19th Annual HP User Conference and Expo

Interex '93

San Francisco, CA ▪ September 19–23, 1993

RETURN TO:
HPL/RESEARCH LIBRARY
BUILDING #2L
P.O. BOX 10490
PALO ALTO, CA. 94303-0971
PHONE # 415-857-3092

VOLUME 1

Proceedings

Sponsored by **Interex,** The International Association of Hewlett-Packard Computer Users

QA76.8
H19 H187
1993
v. 1

Proceedings

Volume 1 of the

19th Annual HP User Conference and Expo

Interex '93

in
San Francisco, CA
September 19–23, 1993

HPL/RESEARCH LIBRARY
BUILDING #2L
P.O. BOX 10490
PALO ALTO, CA 94303-0971

Interex The International Association of Hewlett-Packard Computer Users

Index by Paper Number

K001 Technotrends: Going Beyond Your Competition
Daniel Burrus - Burrus Research Associates, Inc.
P005 Rightsizing: The Key Technology and Business Choices We Face
John R. Logan - Aberdeen Group, Inc.
1000 Using RTE More Effectively in an Increasingly non-RTE HP World
Stephen Gauss - U.S. Naval Observatory
1001 Transferring CPLOT and LPLOT Graphics from the HP 1000 to a PC
Gerald Lisowski - Zeneca, Inc.
1002 Golf and the HP1000
Dave Medlicott - Hewlett-Packard Co.
1003 Managing Multiple Identical RTE-A Systems: A Customized Approach
Larry Ridgley - Hewlett-Packard Co.
1005 Using Modems on the HP 1000 A Series Computers
Alan Tibbetts - Consultant
1006 Standards-Based Networking Services on the HP 1000
Lynn Rodoni, Mydung Tran - Hewlett-Packard Co.
1007 An HP-UX Compatible Spooler for RTE
Todd Poynor - Hewlett-Packard Co.
1008 The HP-RT Real-Time Operating System
Kevin Morgan - Hewlett-Packard Co.
1010 Managing PA RISC Machines for Real-Time Systems.
George Anzinger - Hewlett-Packard Co.
1011 RTE to UNIX Migration Tools and Techniques
Robert Combs - Combs International
1013 Stump the RTE Experts
Alan Tibbetts - Consultant
1014 MEF Users - Should I consider an Upgrade?
Esther Heller - Hewlett-Packard Co.
2000 Remote Network Monitoring and Fault Isolation Using OpenView
Jeff Hodges - Hewlett-Packard Co.
2001 Evaluating System Capability: IBM to HP-UX Benchmark Case Study
Raymond Riedel - Hewlett-Packard Co.
2002 Unlocking the Secrets of UNIX Security
Donna Borsani - Hewlett-Packard Co.
2004 Unix Panic? Don't Panic!
Dennis McClure - HP - North American Response Center
2006 Migrating to a Client/Server Computing Architecture One Step at a Time
Patricia O'Brien - Hewlett-Packard Co.
2007 Why Did My Backup Fail?
Wolfgang Friedrich - Hewlett-Packard Co.
2008 AWK Programming
David Totsch - La Salle National Bank
2009 High Speed Network Transition
Mohammad Malik, Sam Sudarsanam - Hewlett-Packard Co.
2010 Measuring and Monitoring Business Transactions in an Open Systems Environment
Jim Grant - Hewlett-Packard Co.
2011 Meeting Customer's Security Requirements A Crypto
Michael Ferachulou - Hewlett-Packard Co.

Index by Paper Number

2012 Giving End Users Access to Corporate Data
Ron Zambonini - Cognos
3000 Setting a Desktop Strategy: Workstations as Desktops for Business Professionals
Janet M. Muto - Hewlett-Packard Co.
3001 Developing Cross-Platform Multimedia Applications Using Icon Author
Leo Lucas - AimTech Corporation
3002 HP Task Broker Release 1.1
Renato Assini - Hewlett-Packard
3005 Successful Implementation of Software Configuration Management
Tom Burdon - Softool Corporation
3006 COBOL: The Move to the Desktop
Colin Bodell - Micro Focus
3007 Computational Clusters from Hewlett- Packard Company
Diana Headrick - Hewlett Packard
3010 Evaluating X on the PC
David Marks - Walker, Richer & Quinn
4001 Living with Open System Standards: How to Survive Acronym Shock
Rikki Kirzner - Dataquest
4002 The Special Requirements of Commercial UNIX Data Centers
Tom Harris - Operations Control Systems
4003 Getting Your Money's Worth Out of Network Computing
Dave Mahler - Remedy Corporation
4004 Delivering Multimedia in a Networked Environment
Frank Recchia - Hewlett-Packard Co.
4005 Introduction to Network Management
Roger McKee - Consultant
4006 Approaches to the Open System Transition
Todd Hutto - Dun & Bradstreet Software
4008 Emerging Trends in Client/Server Development
Paul Cubbage - Dataquest
4009 Porting Applications Between Unix and DOS Using X-windows
Don Dailey - Quarterdeck Office Systems
4011 68K Workstation Operating System Strategy
Doug Blackwood - Hewlett Packard
4012 SoftBench Framework and ToolTalk: A Technical Comparison
Judy Walker - Hewlett-Packard Co.
4013 X For the Non-Expert: A Tutorial on The X Window System
David Harris - The X Business Group, Inc.
4014 Tuning the HP-UX Series 800 File System
Glen Johnson - CSI Computer Solutions, Inc.
4015 Real-World Unix for Open System Enterprise Document Management
David Weinberger - Interleaf, Inc.
4016 Overview of DME
Andrew McCasker - Systems Center
4017 X Application Performance on Client/Server Systems
Ken Oliver - Hewlett-Packard Co.
4019 UNIX: The Universal Gateway For Cross- Platform Communications
Gwen Peterson - Clarity Software

4020 Analysis of a Mainframe Replacement Project
Gary Gagliardi - FourGen Software, Inc.
4021 The Business Case for Open Systems
Bob Lewin - XOpen
4022 SNMP: What Can It Do for Me?
Andrew J. Phillips - Hewlett-Packard Co.
4023 Commercial Client/Server: How to Develop Business Solutions
Ron Rolland - Anderson Consulting
4024 Getting Ready for the New Standards in Distributed Systems Management
Mike O'Rourke - Tivoli Systems, Inc.
4025 The Open Systems Decision
Jesse Bornfreund - UNIX International
4026 Adaptive Applications for a Client/Server Environment
Tom Axbey - Applix, Inc.
4027 Future Directions for OSF User Environment Technology
Cathy Betz, Vania Joloboff - Open Software Foundation
4028 National Language Support: Providing Data Processing Applications for the Global Marketplace
Natasha Flaherty - Oracle Corporation
4029 Business Issues of EDI: 12 Steps to Successful EDI Implementation
Phyllis Sokol - Sterling Software
4030 Effective Management of HP-UX Systems: Problems and Solutions
Carla Fitzgerald, Alan Paller - Computer Associates, International
4031 Object Database and 4GLs - A Paradigm Shift or a Paradigm Transition
Michael Consoli, Marlene Nesson - Information Builders, Inc.
4032 Implementing HP 9000-890 Platforms: A Case Study
Sameul C. Ellis - Portland Community College
4033 Client/Server for Neophytes
Bob Petrovic - Speedware Corp.
4034 Downsizing to Hewlett-Packard Platforms
John Woolsoncroft - Concepts Dynamic, Inc.
4035 HP Programmer's Toolset: New HP-UX Software Development Tools
Valerie Ho Gibson - Hewlett-Packard Co.
4036 Designing For Usability
Scott McGregor - Prescient Software
4037 Performance and Capacity Management of Distributed UNIX Systems: The Glasshouse Approach
Boris Geller - BGS Systems
4039 Systems Optimization vs Migration to Open Systems
Elliott Chuang - Chuang Associates
4040 Multimedia Support for Relational Database Management Systems
Richard Kozicki - Empress Software
4041 Testing System Performance by Emulating the Real World
Duane Dorch - PERFORMIX, Inc.
4042 Orientation to C++
Ray Swartz - Berkeley Decision/Systems
5000 Are You My Priority?
Jeff Odom - Bahlsen, Inc.
5001 Volume Management: A Mainframe Implementation
Steven Cole - Northern Telecom

Index by Paper Number

5002 MPE/iX Command File Tips and Techniques
Jack Bailie - Exxon Company, U.S.A.
5004 RF/DC and the HP 3000
Joe Howell - Professional Products
5005 Software Quality and the HP 3000
Bob Mead - Hewlett-Packard Co.
5006 The MPE iX Architected Interface Facility--A Case Study Overview
Lee Courtney - Monterey Software Group
5007 Past, Present and Future of Upgrading MPE iX
Tad Olson, et al - Hewlett-Packard
5008 SQL: The Outside Story
Alfredo Rego - Adager
5010 Overview of Image SQL
Jim Sartain - Hewlett-Packard Co.
5011 A Programmer Looks at MPE/iX
Kevin Cooper - Hewlett-Packard
5012 How Did Image Become Relational?
Brad Tashenberg - Bradmark Technolgies, Inc.
5013 Optimizing the Usability of HP OpenView System Manager: A Usability Engineering Case Study
Christine N. Gandel - Hewlett-Packard Co.
5014 Migrating PowerHouse Applications to UNIX
Ben Foulkes - Cognos
5016 System Management MPE/iX Backup Review and Implementing New Technology Overview
Al Dulaney - Hewlett-Packard Co.
5017 How Messy is My Database
David Greer - Robelle Consulting Ltd.
5018 Client/Servier for MPE -- The Sequel
Joseph Geiser - Insurance Data Processing, Inc.
5019 Getting Started with ALLBASE/SQL
John Schmid - 3M Company
5020 I'm Confused! What's the Difference Between Image and Relational Databases?
Brad Tashenberg - Bradmark Technolgies, Inc.
5021 101 (More or Less) Moral Things to Do with HP Susan and the Other MPE/iX Predefined Variables in the Harem.
David Largent - N.G.Gilbert Corp.
5022 Securing a DOE Installation
George Malcolm - Brookhaven National Laborabory
5023 Performance Issues With Large Disks
Sam Yamakoshi - Hewlett-Packard
5024 Using Software to Automate Technical Support
Ken Robertson - Robelle Consulting Ltd.
5025 A Prototype for an MPE/iX Menu-Based User Interface
Mark Farzan - Santa Fe Drilling Co.
5026 Accessing TurboImage From Your PC
Todd Hubbard, Ed Roden - Crowe Chizek
5027 Processing and Integration of Business Information from Decentralized Operations
Miguel Cooper, et al - Industrias Resistol, S.A.
5030 A Primer on Software Objects
Jim Wowchuk - Vanguard Computer Services

5031 English 1A -- A DP Professional's Guide to Writing
Pamela Dickerson - ECI Computer, Inc.
5032 Managing HP3000's Disc Space
Vladimir Volokh - Vesoft
5033 HP3000 to PC Interoperability
Nick Demos - Performance Software Corporation
5034 Working with Native Mode Spooling
Royden Somerville - University of Notre Dame
5035 Designing an INTRINSIC Procedure in Your Favorite Language
Craig Nickerson - United Electric Controls Company
5037 What's New with MPE V?
Bob Holdsworth - Hewlett-Packard Co.
5038 Centralized Enterprise Management with HP OpenView System Manager
Diane Bassett - Hewlett-Packard Co.
5039 STREAMX as a Second Language
Scott DeChant - MEDSTAT Systems, Inc.
5040 What's in a Name Anyhow?
Gordon Gavin - Johnson Hill Press
5042 Secrets of MPE V Tables III: Square Pegs for Round Holes
Craig Nickerson - United Electric Controls Company
5043 How to Win at Your Next EDP Audit!
Tom Harris - Operations Control Systems
5045 It's Hard to Remember Your Objective When Your'e Up to Your @#$ in Auditors
Michael Lam -
5046 Some Useful HP 3000 System Management Techniques
Gilles Schipper - G. Schipper & Associates Inc.
6000 Introduction to Object-Oriented Technologies
Orly Larson - Hewlett-Packard Co.
6001 Incorporating Dynamic Electronic Forms into HP 3000 Applications
Ross G. Hopmans - Brant Technologies, Inc.
6003 The Systems Management Services Partnership
Kyle Adler - Hewlett-Packard Co.
6004 The Myths and Realities of Client-Server Application Development in the 90s
David W. Haberman - Speedware Corp.
6005 Thinking Relational
Alan Camburn - Richard Irwin Associates, Ltd.
6006 DSIS -- Promoting Information Standards for Open Systems Services
Bill Bowman - Hewlett-Packard Co.
6007 Using ALLBASE for High End, Mainframe- Class Production Applications
Vish Krishnan, et al - Hewlett-Packard Co.
6008 Client/Server Application Design Using Graphical User Interfaces and Distributed Object Technology
Tim Ryan - Hewlett-Packard Co.
6009 Integrated Imaging in Financial Systems
Adam Thier - Computron Technologies Corp.
6010 Client/Server: The Best Architecture for Business Applications
Gia Knaus - PeopleSoft, Inc.
6011 Improving the Efficiency and Accuracy of Your Information Capture Through Automated Data Collection
Ray Agrusti - Eagle Consulting & Development Corp.

Index by Paper Number

6012 Moving to Open Systems with a 4GL
Billy S. Hollis - Zortec
6013 Information Integration -- Issues and Approaches
Frank Quemada - Hewlett-Packard Co.
6014 Using Application Response Times as a Metric for Service-Level Agreements
Doug McBride - Hewlett-Packard Co.
6015 Integrating EDI into your Business
Trevor Richards - M.B. Foster Associates, Inc.
6016 Exploring HP-UX: An Investigation Into its Performance Envelope Especially For MPE/iX Fans
Robert Lund - Lund Performance Solutions
6017 Do We Need Transaction monitors on Open Systems?
Rolf Brandt - Infosoft GmbH
6019 Business Use of the Internet
Richard Tinker - Hewlett-Packard Co.
6020 How to Manage Change When Making a Mainframe Alternative Decision
Peggy Parskey - Hewlett-Packard Co.
6021 HP Predictive Support File Transfer: Case Studies in Implementation
Jim Rice - Hewlett-Packard Co.
6024 PowerHouse: The Technology Behind the 4GL
Bob Gibb - Cognos
6025 Why and How You Should Move to SQL
Dave Wiseman - Proactive Systems
7000 Mainframe Migration Alternatives
David Rubinstein - Innovative Information Systems
7001 Distributed Transaction Processing: CICS, Encina and DCE on HP Computers
George Stachnik - Hewlett-Packard Co.
7002 Economic Justification for a System Upgrade
Narendra Sharma - Meriter Hospital, Inc.
7003 SharePlex/iX: HP3000 Clustering Solution
Bryan Dean - Hewlett-Packard Co.
7004 Printing Checks on LaserJets
Debra Canfield - Dairylea Cooperative Inc.
7005 Network to Support Change
Robert Hillseth - Epson America
7006 Data Archiving with Optical Technology
Husni S. Sayed, et al - IEM, Inc.
7007 Programming for POSIX Compliance
Steve Rajavouri - O'Pin Systems
7008 HP's Use of Business Workstations
Andrew J. Phillips - Hewlett-Packard Co.
7009 Guidelines for Writing DCE Applications in COBOL
Charles Knouse - Hewlett-Packard Co.
7010 Project Management and 4GL Methodologies
Suzanne Harmon -
7011 95LX: Tips, Tricks and Talents
John L. Vandegrift - Hewlett-Packard Co.
7012 Developing HP3000 Software Using a UNIX Workstation
Ralph Carpenter - Hewlett-Packard Co.

7013 LAN Migration Strategies

Paul Morgan-Witts - Hewlett-Packard

7014 How-to's of Presentations

Kathy Lee Robertson - Va Department of Housing & Comm. Develop

7015 Exploring Client/Server Computing

Robert Davia - Hewlett-Packard Co.

7016 HP OpenODB in the Oil and Gas Industry

Douglas Dedo - Hewlett-Packard Co.

7017 Meeting the Networking Challenge: Connecting Your PCs, Hosts and LAN Services

Karl Crabs -

7018 Network Planners: Is High-Speed Networking in Your Future?

John Selep - Hewlett-Packard Co.

7019 Client/Server with ALLBASE/SQL and IMAGE/SQL

Bryan Carroll, et al - Hewlett-Packard Co.

7020 Troubleshooting Client/Server Applications

Steven L Adams - Hewlett-Packard Co.

7021 Developing Reliable Systems at Minimal Cost -- A Case Study

Joe A. Sitver - Martin Marietta Energy Systems, Inc.

7022 Networked Computing

Richard Ponschock - Revlon

7023 Breaking Down the Client/Server Barriers Integrating MPE/iX, Posix, Netware/iX and AppleTalk/iX

Bill Lund, Cathy Gunn - Hewlett-Packard Co.

7024 CONQUEST - An Object Oriented HP Configuration and Quote System

Bob Lewis, Susan Stavish - Hewlett-Packard

7025 SCSI: A View From Both Ends of the Cable

Michael Rusnack - Hewlett-Packard Co.

7026 An Innovative Solution for Monitoring the Grouting Operation at Waddell Dam

Mike Lemanski, et al - Hewlett-Packard Co.

7027 Systems Administration - MPE to HP-UX

Ann McDermott - Information Builders

7028 The Management Consultant's Toolbox, Don't Be Left Without It

Leonard Block - The Apex Group

7029 Creating a Low-Cost Client-Server Application

Mark Halstead - Aircast, Inc.

7030 What's So Hard About Software on CD?: A CD-ROM Information Publishing Primer

Katherine Armstrong - Hewlett-Packard Co.

7031 EDI Processing Environments

Richard Peasley - EDI Solutions Inc.

7032 Rightsizing Your Mainfram -- Performance Criteria

James Hepler - Hewlett-Packard Co.

7034 Developing Systems with Your Users as Well as for Your Users

Pamela Herbert - Octet Consulting, Inc.

Adams, Steven L — Troubleshooting Client/Server Applications
7020, Hewlett-Packard Co.
Adler, Kyle — The Systems Management Services Partnership
6003, Hewlett-Packard Co.
Agrusti, Ray — Improving the Efficiency and Accuracy of Your Information Capture Through Automated Data Collection
6011, Eagle Consulting & Development Corp.
Anzinger, George — Managing PA RISC Machines for Real-Time Systems.
1010, Hewlett-Packard Co.
Armstrong, Katherine — What's So Hard About Software on CD?: A CD-ROM Information Publishing Primer
7030, Hewlett-Packard Co.
Assini, Renato — HP Task Broker Release 1.1
3002, Hewlett-Packard
Axbey, Tom — Adaptive Applications for a Client/Server Environment
4026, Applix, Inc.
Bailie, Jack — MPE/iX Command File Tips and Techniques
5002, Exxon Company, U.S.A.
Bassett, Diane — Centralized Enterprise Management with HP OpenView System Manager
5038, Hewlett-Packard Co.
Betz, Cathy, Joloboff, Vania — Future Directions for OSF User Environment Technology
4027, Open Software Foundation
Blackwood, Doug — 68K Workstation Operating System Strategy
4011, Hewlett Packard
Block, Leonard — The Management Consultant's Toolbox, Don't Be Left Without It
7028, The Apex Group
Bodell, Colin — COBOL: The Move to the Desktop
3006, Micro Focus
Bornfreund, Jesse — The Open Systems Decision
4025, UNIX International
Borsani, Donna — Unlocking the Secrets of UNIX Security
2002, Hewlett-Packard Co.
Bowman, Bill — DSIS -- Promoting Information Standards for Open Systems Services
6006, Hewlett-Packard Co.
Brandt, Rolf — Do We Need Transaction monitors on Open Systems?
6017, Infosoft GmbH
Burdon, Tom — Successful Implementation of Software Configuration Management
3005, Softool Corporation
Burrus, Daniel — Technotrends: Going Beyond Your Competition
K001, Burrus Research Associates, Inc.
Camburn, Alan — Thinking Relational
6005, Richard Irwin Associates, Ltd.
Canfield, Debra — Printing Checks on LaserJets
7004, Dairylea Cooperative Inc.
Carpenter, Ralph — Developing HP3000 Software Using a UNIX Workstation
7012, Hewlett-Packard Co.
Carroll, Bryan, et al — Client/Server with ALLBASE/SQL and IMAGE/SQL
7019, Hewlett-Packard Co.
Chuang, Elliott — Systems Optimization vs Migration to Open Systems
4039, Chuang Associates

Cole, Steven Volume Management: A Mainframe Implementation
5001, Northern Telecom
Combs, Robert RTE to UNIX Migration Tools and Techniques
1011, Combs International
Consoli, Michael, Nesson, Marlene Object Database and 4GLs - A Paradigm Shift or a Paradigm Transition
4031, Information Builders, Inc.
Cooper, Kevin A Programmer Looks at MPE/iX
5011, Hewlett-Packard
Cooper, Miguel, et al Processing and Integration of Business Information from Decentralized Operations
5027, Industrias Resistol, S.A.
Courtney, Lee The MPE iX Architected Interface Facility--A Case Study Overview
5006, Monterey Software Group
Crabs, Karl Meeting the Networking Challenge: Connecting Your PCs, Hosts and LAN Services
7017,
Cubbage, Paul Emerging Trends in Client/Server Development
4008, Dataquest
Dailey, Don Porting Applications Between Unix and DOS Using X-windows
4009, Quarterdeck Office Systems
Davia, Robert Exploring Client/Server Computing
7015, Hewlett-Packard Co.
DeChant, Scott STREAMX as a Second Language
5039, MEDSTAT Systems, Inc.
Dean, Bryan SharePlex/iX: HP3000 Clustering Solution
7003, Hewlett-Packard Co.
Dedo, Douglas HP OpenODB in the Oil and Gas Industry
7016, Hewlett-Packard Co.
Demos, Nick HP3000 to PC Interoperability
5033, Performance Software Corporation
Dickerson, Pamela English 1A -- A DP Professional's Guide to Writing
5031, ECI Computer, Inc.
Dorch, Duane Testing System Performance by Emulating the Real World
4041, PERFORMIX, Inc.
Dulaney, Al System Management MPE/iX Backup Review and Implementing New Technology Overview
5016, Hewlett-Packard Co.
Ellis, Sameul C. Implementing HP 9000-890 Platforms: A Case Study
4032, Portland Community College
Farzan, Mark A Prototype for an MPE/iX Menu-Based User Interface
5025, Santa Fe Drilling Co.
Ferachulou, Michael Meeting Customer's Security Requirements A Crypto
2011, Hewlett-Packard Co.
Fitzgerald, Carla, Paller, Alan Effective Management of HP-UX Systems: Problems and Solutions
4030, Computer Associates, International
Flaherty, Natasha National Language Support: Providing Data Processing Applications for the Global Marketplace
4028, Oracle Corporation
Foulkes, Ben Migrating PowerHouse Applications to UNIX
5014, Cognos
Friedrich, Wolfgang Why Did My Backup Fail?
2007, Hewlett-Packard Co.

Gagliardi, Gary — Analysis of a Mainframe Replacement Project
4020, FourGen Software, Inc.

Gandel, Christine N. — Optimizing the Usability of HP OpenView System Manager: A Usability Engineering Case Study
5013, Hewlett-Packard Co.

Gauss, Stephen — Using RTE More Effectively in an Increasingly non-RTE HP World
1000, U.S. Naval Observatory

Gavin, Gordon — What's in a Name Anyhow?
5040, Johnson Hill Press

Geiser, Joseph — Client/Servier for MPE -- The Sequel
5018, Insurance Data Processing, Inc.

Geller, Boris — Performance and Capacity Management of Distributed UNIX Systems: The Glasshouse Approach
4037, BGS Systems

Gibb, Bob — PowerHouse: The Technology Behind the 4GL
6024, Cognos

Grant, Jim — Measuring and Monitoring Business Transactions in an Open Systems Environment
2010, Hewlett-Packard Co.

Greer, David — How Messy is My Database
5017, Robelle Consulting Ltd.

Haberman, David W. — The Myths and Realities of Client-Server Application Development in the 90s
6004, Speedware Corp.

Halstead, Mark — Creating a Low-Cost Client-Server Application
7029, Aircast, Inc.

Harmon, Suzanne — Project Management and 4GL Methodologies
7010,

Harris, David — X For the Non-Expert: A Tutorial on The X Window System
4013, The X Business Group, Inc.

Harris, Tom — The Special Requirements of Commercial UNIX Data Centers
4002, Operations Control Systems

Harris, Tom — How to Win at Your Next EDP Audit!
5043, Operations Control Systems

Headrick, Diana — Computational Clusters from Hewlett- Packard Company
3007, Hewlett Packard

Heller, Esther — MEF Users - Should I consider an Upgrade?
1014, Hewlett-Packard Co.

Hepler, James — Rightsizing Your Mainfram -- Performance Criteria
7032, Hewlett-Packard Co.

Herbert, Pamela — Developing Systems with Your Users as Well as for Your Users
7034, Octet Consulting, Inc.

Hillseth, Robert — Network to Support Change
7005, Epson America

Ho Gibson, Valerie — HP Programmer's Toolset: New HP-UX Software Development Tools
4035, Hewlett-Packard Co.

Hodges, Jeff — Remote Network Monitoring and Fault Isolation Using OpenView
2000, Hewlett-Packard Co.

Holdsworth, Bob — What's New with MPE V?
5037, Hewlett-Packard Co.

Hollis, Billy S. — Moving to Open Systems with a 4GL
6012, Zortec

Hopmans, Ross G. Incorporating Dynamic Electronic Forms into HP 3000 Applications
6001, Brant Technologies, Inc.
Howell, Joe RF/DC and the HP 3000
5004, Professional Products
Hubbard, Todd, Roden, Ed Accessing TurboImage From Your PC
5026, Crowe Chizek
Hutto, Todd Approaches to the Open System Transition
4006, Dun & Bradstreet Software
Johnson, Glen Tuning the HP-UX Series 800 File System
4014, CSI Computer Solutions, Inc.
Kirzner, Rikki Living with Open System Standards: How to Survive Acronym Shock
4001, Dataquest
Knaus, Gia Client/Server: The Best Architecture for Business Applications
6010, PeopleSoft, Inc.
Knouse, Charles Guidelines for Writing DCE Applications in COBOL
7009, Hewlett-Packard Co.
Kozicki, Richard Multimedia Support for Relational Database Management Systems
4040, Empress Software
Krishnan, Vish, et al Using ALLBASE for High End, Mainframe- Class Production Applications
6007, Hewlett-Packard Co.
Lam, Michael It's Hard to Remember Your Objective When Your'e Up to Your @#$ in Auditors
5045,
Largent, David 101 (More or Less) Moral Things to Do with HP Susan and the Other MPE/iX Predefined Variables in the Harem.
5021, N.G.Gilbert Corp.
Larson, Orly Introduction to Object-Oriented Technologies
6000, Hewlett-Packard Co.
Lemanski, Mike, et al An Innovative Solution for Monitoring the Grouting Operation at Waddell Dam
7026, Hewlett-Packard Co.
Lewin, Bob The Business Case for Open Systems
4021, XOpen
Lewis, Bob, Stavish, Susan CONQUEST - An Object Oriented HP Configuration and Quote System
7024, Hewlett-Packard
Lisowski, Gerald Transferring CPLOT and LPLOT Graphics from the HP 1000 to a PC
1001, Zeneca, Inc.
Logan, John R. Rightsizing: The Key Technology and Business Choices We Face
P005, Aberdeen Group, Inc.
Lucas, Leo Developing Cross-Platform Multimedia Applications Using Icon Author
3001, AimTech Corporation
Lund, Bill, Gunn, Cathy Breaking Down the Client/Server Barriers Integrating MPE/iX, Posix, Netware/iX and AppleTalk/iX
7023, Hewlett-Packard Co.
Lund, Robert Exploring HP-UX: An Investigation Into its Performance Envelope Especially For MPE/iX Fans
6016, Lund Performance Solutions
M. Muto, Janet Setting a Desktop Strategy: Workstations as Desktops for Business Professionals
3000, Hewlett-Packard Co.
Mahler, Dave Getting Your Money's Worth Out of Network Computing
4003, Remedy Corporation
Malcolm, George Securing a DOE Installation
5022, Brookhaven National Laborabory

Malik, Mohammad, Sudarsanam, Sam 2009, Hewlett-Packard Co.	High Speed Network Transition
Marks, David 3010, Walker, Richer & Quinn	Evaluating X on the PC
McBride, Doug 6014, Hewlett-Packard Co.	Using Application Response Times as a Metric for Service-Level Agreements
McCasker, Andrew 4016, Systems Center	Overview of DME
McClure, Dennis 2004, HP - North American Response Center	Unix Panic? Don't Panic!
McDermott, Ann 7027, Information Builders	Systems Administration - MPE to HP-UX
McGregor, Scott 4036, Prescient Software	Designing For Usability
McKee, Roger 4005, Consultant	Introduction to Network Management
Mead, Bob 5005, Hewlett-Packard Co.	Software Quality and the HP 3000
Medlicott, Dave 1002, Hewlett-Packard Co.	Golf and the HP1000
Morgan, Kevin 1008, Hewlett-Packard Co.	The HP-RT Real-Time Operating System
Morgan-Witts, Paul 7013, Hewlett-Packard	LAN Migration Strategies
Nickerson, Craig 5035, United Electric Controls Company	Designing an INTRINSIC Procedure in Your Favorite Language
Nickerson, Craig 5042, United Electric Controls Company	Secrets of MPE V Tables III: Square Pegs for Round Holes
O'Brien, Patricia 2006, Hewlett-Packard Co.	Migrating to a Client/Server Computing Architecture One Step at a Time
O'Rourke, Mike 4024, Tivoli Systems, Inc.	Getting Ready for the New Standards in Distributed Systems Management
Odom, Jeff 5000, Bahlsen, Inc.	Are You My Priority?
Oliver, Ken 4017, Hewlett-Packard Co.	X Application Performance on Client/Server Systems
Olson, Tad, et al 5007, Hewlett-Packard	Past, Present and Future of Upgrading MPE iX
Parskey, Peggy 6020, Hewlett-Packard Co.	How to Manage Change When Making a Mainframe Alternative Decision
Peasley, Richard 7031, EDI Solutions Inc.	EDI Processing Environments
Peterson, Gwen 4019, Clarity Software	UNIX: The Universal Gateway For Cross- Platform Communications
Petrovic, Bob 4033, Speedware Corp.	Client/Server for Neophytes
Phillips, Andrew J. 7008, Hewlett-Packard Co.	HP's Use of Business Workstations

Phillips, Andrew J. 4022, Hewlett-Packard Co.	SNMP: What Can It Do for Me?
Ponschock, Richard 7022, Revlon	Networked Computing
Poynor, Todd 1007, Hewlett-Packard Co.	An HP-UX Compatible Spooler for RTE
Quemada, Frank 6013, Hewlett-Packard Co.	Information Integration -- Issues and Approaches
Rajavouri, Steve 7007, O'Pin Systems	Programming for POSIX Compliance
Recchia, Frank 4004, Hewlett-Packard Co.	Delivering Multimedia in a Networked Environment
Rego, Alfredo 5008, Adager	SQL: The Outside Story
Rice, Jim 6021, Hewlett-Packard Co.	HP Predictive Support File Transfer: Case Studies in Implementation
Richards, Trevor 6015, M.B. Foster Associates, Inc.	Integrating EDI into your Business
Ridgley, Larry 1003, Hewlett-Packard Co.	Managing Multiple Identical RTE-A Systems: A Customized Approach
Riedel, Raymond 2001, Hewlett-Packard Co.	Evaluating System Capability: IBM to HP-UX Benchmark Case Study
Robertson, Kathy Lee 7014, Va Department of Housing & Comm. Develop	How-to's of Presentations
Robertson, Ken 5024, Robelle Consulting Ltd.	Using Software to Automate Technical Support
Rodoni, Lynn, Tran, Mydung 1006, Hewlett-Packard Co.	Standards-Based Networking Services on the HP 1000
Rolland, Ron 4023, Anderson Consulting	Commercial Client/Server: How to Develop Business Solutions
Rubinstein, David 7000, Innovative Information Systems	Mainframe Migration Alternatives
Rusnack, Michael 7025, Hewlett-Packard Co.	SCSI: A View From Both Ends of the Cable
Ryan, Tim 6008, Hewlett-Packard Co.	Client/Server Application Design Using Graphical User Interfaces and Distributed Object Technology
Sartain, Jim 5010, Hewlett-Packard Co.	Overview of Image SQL
Sayed, Husni S., et al 7006, IEM, Inc.	Data Archiving with Optical Technology
Schipper, Gilles 5046, G. Schipper & Associates Inc.	Some Useful HP 3000 System Management Techniques
Schmid, John 5019, 3M Company	Getting Started with ALLBASE/SQL
Selep, John 7018, Hewlett-Packard Co.	Network Planners: Is High-Speed Networking in Your Future?
Sharma, Narendra 7002, Meriter Hospital, Inc.	Economic Justification for a System Upgrade

Sitver, Joe A. 7021, Martin Marietta Energy Systems, Inc.	Developing Reliable Systems at Minimal Cost -- A Case Study
Sokol, Phyllis 4029, Sterling Software	Business Issues of EDI: 12 Steps to Successful EDI Implementation
Somerville, Royden 5034, University of Notre Dame	Working with Native Mode Spooling
Stachnik, George 7001, Hewlett-Packard Co.	Distributed Transaction Processing: CICS, Encina and DCE on HP Computers
Swartz, Ray 4042, Berkeley Decision/Systems	Orientation to C++
Tashenberg, Brad 5012, Bradmark Technolgies, Inc.	How Did Image Become Relational?
Tashenberg, Brad 5020, Bradmark Technolgies, Inc.	I'm Confused! What's the Difference Between Image and Relational Databases?
Thier, Adam 6009, Computron Technologies Corp.	Integrated Imaging in Financial Systems
Tibbetts, Alan 1005, Consultant	Using Modems on the HP 1000 A Series Computers
Tibbetts, Alan 1013, Consultant	Stump the RTE Experts
Tinker, Richard 6019, Hewlett-Packard Co.	Business Use of the Internet
Totsch, David 2008, La Salle National Bank	AWK Programming
Vandegrift, John L. 7011, Hewlett-Packard Co.	95LX: Tips, Tricks and Talents
Volokh, Vladimir 5032, Vesoft	Managing HP3000's Disc Space
Walker, Judy 4012, Hewlett-Packard Co.	SoftBench Framework and ToolTalk: A Technical Comparison
Weinberger, David 4015, Interleaf, Inc.	Real-World Unix for Open System Enterprise Document Management
Wiseman, Dave 6025, Proactive Systems	Why and How You Should Move to SQL
Woolsoncroft, John 4034, Concepts Dynamic, Inc.	Downsizing to Hewlett-Packard Platforms
Wowchuk, Jim 5030, Vanguard Computer Services	A Primer on Software Objects
Yamakoshi, Sam 5023, Hewlett-Packard	Performance Issues With Large Disks
Zambonini, Ron 2012, Cognos	Giving End Users Access to Corporate Data

K001
Technotrends: Going Beyond Your Competition

Daniel Burrus
Burrus Research Associates, Inc.
PO Box 26413, Milwaukee, WI 53226-0413
414-774-7790

©1993

1 **GENETIC ENGINEERING** All living organisms are made of cells, and in those cells are genes that have a readable code defining all aspects of the plant or animal. Key tools include Recombinant DNA (rDNA) technology -- the mapping, restructuring, and remodeling of the gene code to eliminate or enhance a specific trait -- and Anti-Sense RNA compounds, which have the power to block the expression of specific genes.

2 **ADVANCED BIOCHEMISTRY** Using advanced biological techniques, biochemists are creating new disease diagnostic systems, highly effective "super drugs," advanced drug delivery systems, and a variety of new bioindustrial applications. Advanced biochemistry techniques were used to create Interleukin-2, a new class of drug that can fight diseases like cancer. Monoclonal antibodies have been produced that bind only to a specific molecule and are used to diagnose disease, pinpoint specific genes, and purify rare substances. Two other examples include fetal-cell transplants, which can be used to treat blood disorders such as sickle cell anemia, Parkinson's disease, diabetes and radiation exposure, and photoactive drugs, which are activated when exposed to light.

3 **DIGITAL ELECTRONICS** Digital devices translate signals into the 0s and 1s that computers understand. The original signal is sampled instant-by-instant, converted to a numerical map, and sent to a receiver. Traditional electronic devices, as well as magnetic and optical devices, can use digital techniques. Key tools include Digital Imaging, Digital Television, Digital Cellular Telephones, and Personal Communication Networks.

4 **OPTICAL DATA STORAGE** Optical memory systems use lasers to read information that is stored in digital form. Examples include *all* optical disks, optical film, floptical disks, and bar code readers. Optical storage devices can randomly access digital information at high speeds. They can contain audio, video, and computer data at the same time. By the late 1990s, most digital information storage will be chip-based or optical-based, allowing for more efficient use of digital data.

5 **ADVANCED VIDEO DISPLAYS** There are two main types of advanced video displays, Advanced Flat-Panel Displays that will provide us with full color, flat, and lightweight television screens in a variety of sizes, and High Definition Television (HDTV), which describes a very high-resolution screen whose dimensions resemble that of a picture in a movie theater. HDTV displays will find their first applications in medicine, advanced simulations, and advanced workstations.

6 **ADVANCED COMPUTERS** Computers are electronic calculating machines that can process information and follow programmed instructions. Advanced computers cover all related hardware and systems that are based on advanced chip technology such as personal computers, supercomputers, and micro and minicomputers. Key tools include Parallel Processing Computers, Multimedia Computers, Electronic Notepads, Telecomputers, and Multi-Sensory Robotics.

7 **DISTRIBUTED COMPUTING** Distributed computing includes both enterprise and multiple enterprise computer integration and the transparent multi-user sharing of information and applications across a multi-vendor computer network that supports day-to-day business activities.

8 **ARTIFICIAL INTELLIGENCE** AI is the capability of a computer to perform functions that are normally attributed to human intelligence such as learning, adapting, recognizing, classifying, reasoning, self-correction, and improvement. Key tools include Expert Systems, Advanced Simulations, Object Oriented Programming, Fuzzy Logic, Neural Networks, Voice Recognition, and Image Processing.

9 **LASERS** The word laser stands for Light Amplification by Stimulated Emission of Radiation. Laser light covers a narrow range of wavelengths, tends to be coherent, and is emitted in a narrow directional beam of high intensity. Laser devices can range in size from a pinhead to the size of a football field. Their light ranges from invisible ultraviolet and infrared through all colors of the rainbow. A wide range of applications are already in use, including eye surgery, compact disks, laser scalpels, and holography.

10 **FIBER OPTICS** Fiber optics provides a digital highway in which photons, particles of light, travel. An optical fiber is a hair-thin strand of glass composed of silicon and other materials with a light transmitting core and a layer of material that keeps the light from straying. When used for communications, they can carry four signals at once: telephone, television, radio, and computer data.

11 **MICROWAVES** Microwaves are electromagnetic waves having a wavelength in the region between infrared and short-wave radio. Currently, microwaves have two major application categories: sending wireless digital information and heating objects by creating molecular movement inside the object. Uses range from microwave clothes dryers and microwave scalpels to heating inoperable cancerous tumors to a temperature that kills the cancer but leaves the healthy cells alive.

12 **ADVANCED SATELLITES** As advanced satellites with diverse uses are put into orbit by more and more countries, they will play an ever increasing role in worldwide government and business communications, as well as in studying, mapping, and surveying the earth. Landstat is, for example, being used for oil and mineral exploration. Navestar can be used to determine exact locations of all forms of transportation on the planet. Surveillance will continue to play a big role for satellites. Key new tools will include Low Earth Orbit (LEO) Satellites and Direct Broadcast Satellites (DBS).

13 **PHOTOVOLTAIC CELLS (PV)** When photons of sunlight strike a solar cell, electrons are knocked free from silicon atoms and are drawn off by a grid of metal conductors. This action yields a flow of electricity (direct current). PV cells require no fuel, are self-contained with no moving parts, are nonpolluting, and have a lifetime of over twenty years. Today's most popular type of PV cell is amorphous (non-crystalline) silicon, which is forty times more light absorbing than crystalline silicon. Currently, PV cells can convert sunlight directly into electricity at the efficiency rate of over twenty-eight percent. They can be used for applications such as pocket calculators, refrigerators, portable communications, and remote and rural electrification.

14 **MICROMECHANICS** Micromechanics involves the designing and building of tiny mechanisms, such as valves, accelerometers, pressure and force sensors, and surgical tools. Micromachines can be etched in batches on silicon wafers and then sliced into separate chips. They can then be linked up with microelectronic circuits and used for applications such as monitoring pollution, aiding medical research, and giving robots a sense of touch.

15 **NEW POLYMERS** Polymers are complex chemical structures that can be combined with reinforcing substances and adapted to many uses. By rearranging loops and chains of carbon, oxygen, hydrogen, and nitrogen, chemists are producing polymers that can conduct electricity, dissolve in sunlight, carry light waves, and function as moving parts in automobiles. Currently, there are over 60,000 different polymers with applications ranging from garbage bags to U.S. Army tanks.

16 **HIGH TECH CERAMICS** Ceramic materials are hard, chemically inert, and resistant to corrosion, wear, and high temperatures. Any substance except carbon-based compounds can be used when making ceramic materials. Most ceramics are electrical insulators and are transparent to most forms of electromagnetic radiation. Applications include abrasives for cutting tools, heat shields, ball bearings, engine components, and artificial bone implants.

17 **FIBER-REINFORCED COMPOSITES** Composites are materials, such as ceramics and plastics, that have been reinforced with synthetic fibers and carbon filaments. Composites are beginning to replace some automobile and airplane parts because they are lightweight, resist corrosion, and are often stronger than steel. For example, Beech Aircraft Corporation's business jet, the Starship, has an all-composite body. Applications range from building materials to bridges.

18 **SUPERCONDUCTORS** Superconductors are materials that carry electricity without any loss of energy. Near-term applications include less expensive but more advanced magnetic imaging machines for hospitals, superconducting TV antennas, faster computer circuits in mainframe computers using thin-film superconductors, and small and efficient electric motors.

19 **THIN-FILM DEPOSITION** A process that deposits layers of a specific material as thin as a single atom onto almost any surface. One process called Chemical Vapor Deposition (CVD) uses a coating material that is heated until it vaporizes and is then allowed to condense into the surface to be coated. Another example, Diamond Thin-Film Coating, is produced by mixing methane and hydrogen under the proper conditions, resulting in a diamond film about a millionth of an inch thick. A third process, Molecular Beam Epitaxy, is a semiconductor fabrication process used to build up devices one molecular layer at a time. This process allows different materials and types of doping to be sandwiched precisely in layers.

20 **MOLECULAR DESIGNING** A technique for creating custom-made materials. Scientists, using a supercomputer, can decide what properties they want a material to have and then, using advanced computer graphics and modeling programs, custom design a new material molecule-by-molecule, atom-by-atom. By using lasers to lay down atoms in a precise pattern on surfaces, molecular designers can alter material properties, such as making metals become glass and insulators become conductors. The first products to move out of the lab are tailor-made enzymes for industry.

Rightsizing: The Key Technology and Business Choices We Face

Interex'93

August 23, 1993

John R. Logan
Vice President
Aberdeen Group, Inc.
92 State Street
Boston, MA 02109
617/723/7890

Aberdeen Notes:
Aberdeen Group, Inc. is a computer and communications research and consulting organization closely monitoring user needs, technological changes, and market developments.

Based on a comprehensive analytical framework, Aberdeen provides fresh insights into the future of computing and its implications for users and the industry.

Aberdeen performs specific projects for a select group of clients requiring strategic and tactical advice and hard answers on how to manage computer technology.

Audience Notes:

Agenda

- **Why The Rush From The Mainframe**
- **Successful Rightsizing ACTION**
- **Comparing HP's Rightsizing Technology With The Other Suppliers**
- **HP-experienced IS Business Role In Rightsizing**

AberdeenGroup

Aberdeen Notes:
Today's presentation will provide a comprehensive viewpoint of how enterprises are rightsizing their information systems from a legacy mainframe past. And since this is a once-a-year opportunity for Interex members to convene as a national group and discuss the state-of-the-industry, Aberdeen will also review how HP's competitors are approaching the rightsizing challenge.

The information that is being presented today is a summary of Aberdeen's on-going research on organizations that are actively implementing rightsizing projects.

Audience Notes:

Why The Mainframe Is Going Away

- The BOSS said so -- a strategic decision
- Line managers demand access to on-line data -- green-stripe reports unacceptable
- Mainframe software fees are increasing with no productivity benefits -- Finance department resists
- HP's midrange computer technology surpasses IBM's mainframe products

AberdeenGroup

Aberdeen Notes:

Enterprise's that began the rightsizing process in the late 1980s began obtaining the financial rewards by the early 1990s – and their competitors are now playing catch-up.

Departmental managers cannot effectively manage their operations through mainframe-generated batch reports with data that is out-of-date and must be re-keyed into decision support applications such as Lotus on a PC. Note that this is a more important impetus to move off the mainframe today than lowering IS costs.

Finally, comparing IBM's mainframe technology to HP's midrange, enterprise decision makers believe that HP has surpassed IBM.

Audience Notes:

Goals For Rightsizing

- **Lower IS costs -- real but insular view**

Management Vision to re-engineer the enterprise for:

- **Lowering overhead (SG&A) costs**
- **Increasing customer satisfaction**
- **Creating new revenue streams**

AberdeenGroup

Aberdeen Notes:
Many rightsizing projects have been initiated as a way of lowering an enterprise's IS costs – typically requiring that these expenses by lowered by 25%-50% over three years with no decrease in service levels. But the reality is that non-technical management's vision is to re-engineer the enterprise to be successful through the rest of the decade.

The term downsizing now refers more to middle-management ranks than computers and this is the primary financial objective of computer rightsizing. But in addition, IS staff is being pushed to use its creativity and imagination to recommend innovate methods for increasing customer satisfaction and creating new revenue streams.

Audience Notes:

Enterprise Financial Issues

BigCo Ltd Income Statement

100% Revenue
55% Cost of Goods Sold (COGS)
35% Selling, General, and Administrative
(IS is 3% of Total Revenue)
10% Profit Before Tax (PBT)

If IS Lowers:

- SG&A costs by 20%, PBT increases 70%
- COGS costs by 5%, PBT increases 30%
- IS costs by 50%, PBT increases 15%

AberdeenGroup

Aberdeen Notes:

IS has limited resources in terms of knowledgeable and capable staff that it can apply to business-oriented rightsizing projects.

Past experience shows that rightsizing projects focusing on how an enterprise operates have the potential to reduce overhead costs by 20% and increase profitability by 70%. The major fear of every CEO is that a competitor will take this approach first and then reduce price to the point that others will be driven into an unprofitable situation.

Merely attempting to lower IS costs through rightsizing is self defeating for an enterprise — the returns are minimal compared to business and process re-engineering.

Audience Notes:

ACTION

Architecture

Components

Timing

Integration

Organization

New economics

AberdeenGroup

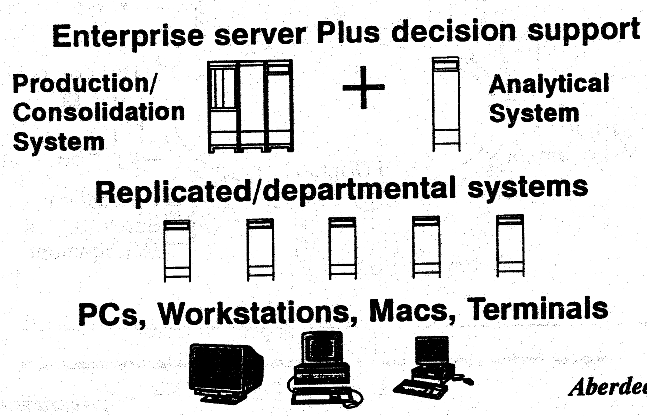
Aberdeen Notes:
To summarize the essential points required to successfully rightsize, Aberdeen has created a simple acronym ACTION.

The vast majority of rightsizing projects that fail do so because IS management does not create a vision of the future with senior business managers. Aberdeen cannot overemphasize how important it is for IS executives to *take charge* of the process to create a rightsized information architecture for their enterprise that automates efficient business processes that provide superior customer satisfaction than the competitors can offer. With this architectural vision as a basis, the technical components, timing, application integration, organizational changes, and new economic realities can pragmatically be derived.

Audience Notes:

Enterprise Topology

Three-tier Plus in a distributed topology is state-of-the art



Enterprise server Plus decision support

Production/ Consolidation System + **Analytical System**

Replicated/departmental systems

PCs, Workstations, Macs, Terminals

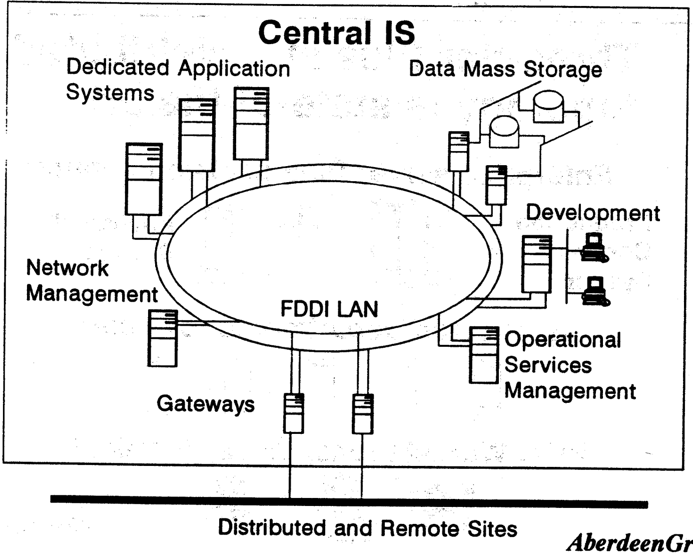
AberdeenGroup

Aberdeen Notes:
Aberdeen's research shows that the three-tier plus architecture meets the needs of both users and IS professionals. Quite frankly, the major technical failures of rightsizing projects have occurred within enterprises that attempted to implement two-tier architectures – today's technology cannot support this overly simplistic approach to rightsizing.

The key to user satisfaction is the emphasis IS puts on providing line middle managers with an analytical systems from which they can access current data to make realtime decisions. And IS can provide remote back-up, data integrity, security, and application version control through HP systems software operated by IS professionals from the enterprise's production system.

Audience Notes:

Rightsized Central IS Architecture


Central IS
Dedicated Application Systems
Data Mass Storage
Development
Network Management
FDDI LAN
Operational Services Management
Gateways
Distributed and Remote Sites

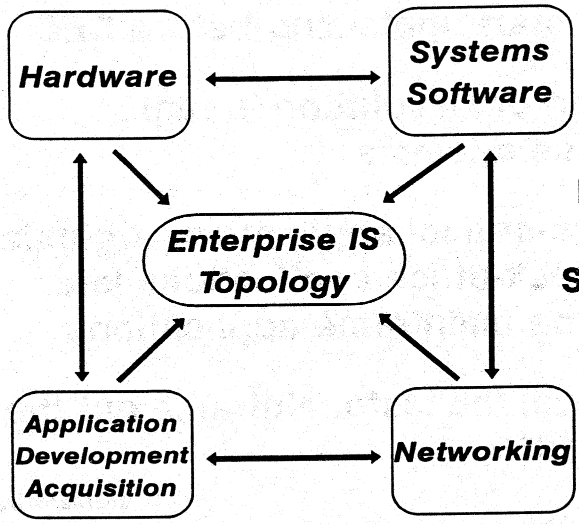
AberdeenGroup

Aberdeen Notes:

Even central IS datacenters are rightsizing by removing the mainframe. In this topology, an application is dedicated to a single midrange system — a very large application can even be spread over multiple midrange systems. This approach lowers software licensing costs and allows IS to upgrade systems by application requirements.

Audience Notes:

Critical Technology Areas



Hardware

Systems Software

Enterprise IS Topology

Application Development Acquisition

Networking

**Multiuser
versus
Single user**

AberdeenGroup

Aberdeen Notes:
Rightsized enterprise architectures are created by blending technology components out of four main groups — hardware, systems software, application development and/or acquisition, and networking. By using open systems components that meet industry standards, enterprises are able to create a dynamic environment that can be improved as part of a continuous process over time.

At the box level, too many non-technical executives are naively confusing the single-user computing they are familiar with with multiuser enterprise computing. The two are very different and believing that PC technology can be used to run a coordinated enterprise will result in disastrous mistakes. And we all need more expertise in networking technology.

Audience Notes:

Timing

- **IS must start rightsizing before CEO**
- **Evolution or Revolution is real enterprise dilemma**
- **Business-critical applications rightsized first -- Back-office applications later Outsource mainframe applications**
- **Plan to roll the last mainframe out the door in 1996**

AberdeenGroup

Aberdeen Notes:
Enterprise management expects IS to be proactive and initiate rightsizing efforts within the enterprise — or it will surmise that the current IS organization is incapable of doing so.

While many enterprises have been attempting to manage to an evolutionary approach to rightsizing off the mainframe, the trend is toward a full revolutionary stance — the business benefits are so compelling and the results of failure to rightsize quickly enough so enormous.

Applications that have the largest impact on the enterprise such as order entry and customer service are typically re-engineered first and then integrated into updated count-the-money back-office applications later.

Audience Notes:

Integration

- **Data conversion is a key issue**
- **Surround the mainframe with operational decision support systems**
- **Asynchronous data transfer -- time posting of data -- among applications is key new dimension**
- **Use open systems components -- today's new applications are tomorrow's legacy**

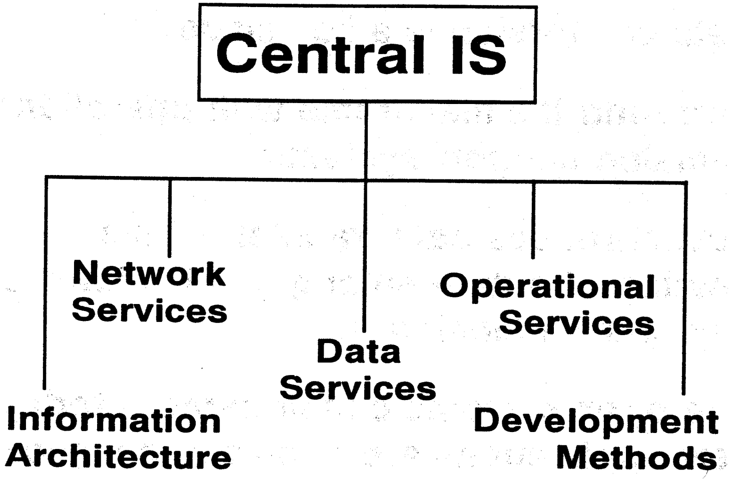
AberdeenGroup

Aberdeen Notes:
Never underestimate the *management* effort required to convert current data into the new rightsized systems. Much legacy data is incorrect and yet organizations have found ways of working around the problem — but moving to new systems brings this hidden issue to the surface.

Rightsized architectures must take into account the requirement to integrate new applications with existing legacy ones that will be transformed at a later date — and the fact that new applications in 1994 will be legacy applications in 1996. The open systems approach is the only method to alleviate the problems of needing to integrate even newer applications into the next-generation ones we are working on today.

Audience Notes:

Organizational Impact



Central IS

Network Services

Data Services

Operational Services

Information Architecture

Development Methods

AberdeenGroup

Aberdeen Notes:
Contrary to what some media headlines proclaim, central IS not going away — but it is evolving to better manage rightsized architectures.

The Information Architecture function is responsible for coordinating the enterprise's strategy with IS' continuously changing capabilities; Network Services monitors performance, detects and corrects faults, and manages program versions; Data Services coordinates personal, enterprise, customer, supplier, electronic, and non-electronic data; Operational Services supports and enhances departmental applications; and Development Methods acts as the technology evaluator and establishes an enterprise-wide development methodology and practices.

Audience Notes:

Organizational Impact

Line	MIS
CEO	CIO
Middle Managers	Director of MIS
Staff	MIS Staff

AberdeenGroup

Aberdeen Notes:
There are six major groups with the organization that will be impacted by the transformation of IS to a rightsized environment. IS executives should ideally meet the requirements of each of these groups to ensure a smooth transition.

Of particular concern are the CEO who believes today that establishing the proper IS infrastructure is a significant part of this most senior job function; line middle managers who are demanding better IS tools to do their jobs more effectively; MIS staff, especially mainframe trained staff, that know how the enterprise works but do not know the tools of rightsized systems; and the CIO who is under increasing pressure to make business decisions, not merely technical.

Audience Notes:

New Economics

- **Maximize total enterprise profitability and effectiveness -- not just IS**
- **Prioritize IS functions by economic return to enterprise**
- **Only upgrade technical components when economic returns justify**
- **Continuously evaluate ways IS can improve business processes -- in all areas**

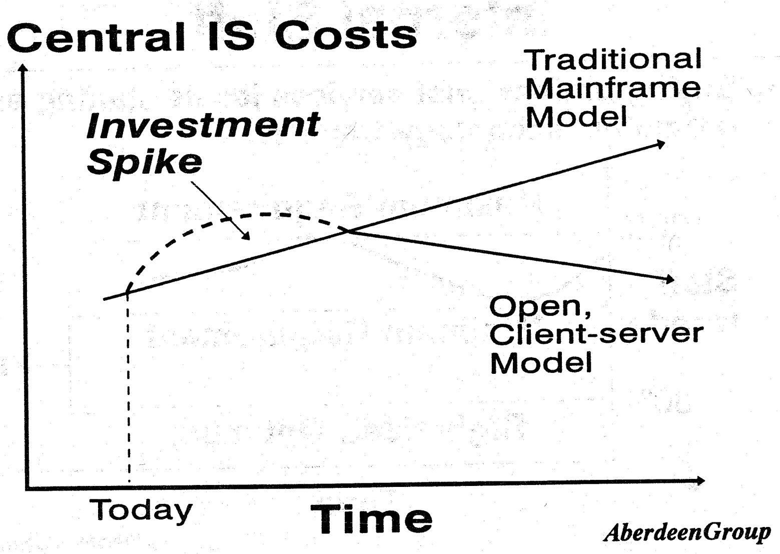
AberdeenGroup

Aberdeen Notes:
The mainframe IS shop is viewed by senior line management as a cost center that is a drain on the enterprise's resources — the old economics. The New Economics focuses on the concept that an investment in IS will provide positive returns — IS is a way to lower costs and make money.

Almost all enterprises that have successfully rightsized report that many of the greatest benefits came from IS executives who reviewed operational aspects of the business with the intention of improving effectiveness through the flexibility rightsized systems provide. In the new economics, IS executives take even greater responsibility for the financial strength of their entire enterprise.

Audience Notes:

Minimizing the Investment Spike



Aberdeen Notes:
The cost justification for investing in rightsized systems is that open, client-server systems will lead to lower IS costs than the traditional mainframe model.

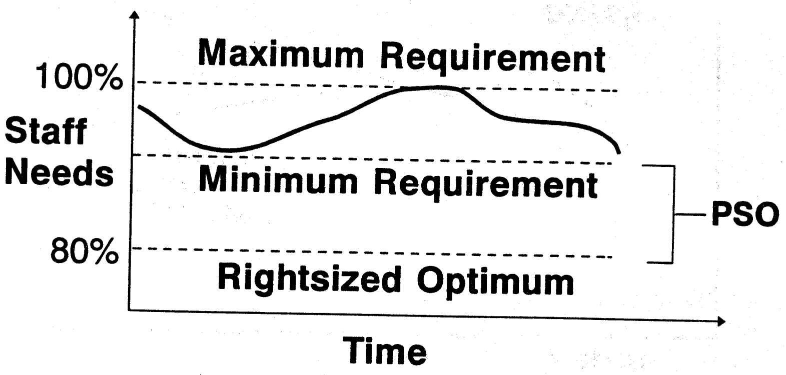
Investment costs required include training, hardware, software, and professional services. The investment spike can be best leveled though the use of leasing vehicles.

Many enterprises that have actively pursued the rightsizing process for several years report IS costs do not decline — the enterprise finds the return of increased investments justifies expanded spending to both increase profitability and gain a competitive advantage.

Audience Notes:

Buying Services versus Internal Staff

- **Buying professional services levels staffing and promotes technology transfer**


Maximum Requirement
100%
Staff Needs
Minimum Requirement
PSO
80%
Rightsized Optimum
Time

AberdeenGroup

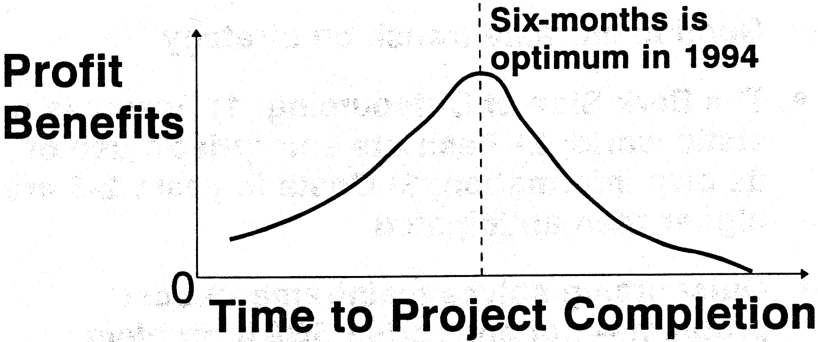
Aberdeen Notes:
Recognizing that managing the efficient transfer of continuously changing technology into internal IS is a key aspect of the rightsizing process, IS executives are staffing their internal organizations at 80% of their maximum personnel needs and using professional service organizations to fill the gap.

The new economics of rightsizing dictates that no IS organization is big enough and smart enough to do everything itself. Therefore, professional service organizations (PSOs) are used to augment internal IS staff. The PSOs act as a technology catalyst for change and improvement.

Audience Notes:

Buying Services versus Internal Staff

- **Buying professional services levels to hit target completion dates for new applications**



AberdeenGroup

Aberdeen Notes:
One of the major reasons for moving to a rightsized IS topology is to put applications into production within a reasonable time. Quite frankly, major mainframe applications typically take 18-30 months to implement after approval. Senior line management finds this unacceptable.

Under the new economics, it appears that six months is the optimum time between when major new applications are approved and when they are put into production. Less that six months does not seen to provide adequate time for business planning processes — after six months and the business environment has changed. The new economics of rightsizing shows that purchasing PSO services should be used to augment internal IS staff resources to meet the six-month optimum criteria.

Audience Notes:

IS Outsourcing

- **Lowers today's mainframe costs ASAP**
- **Seductive to business managers wanting to meet next quarter's profit goals**
- **Good mainframe transition strategy**
- **The Dark Side of Outsourcing: 1) Assumes a static world; 2) Restricts enterprise's use of its own information; 3) Costs in years 2-5 are higher than anticipated**
- **Outsourcing solves mainframe IS cost problem -- not enterprise SG&A problem**

AberdeenGroup

Aberdeen Notes:
An alternative to mainframe IS rightsizing is simply outsourcing all the enterprise's central data processing resources. While outsourcing is a good mainframe transition strategy to rightsizing, past experience shows that the long-term benefits have proven highly elusive.

Total IS outsourcing may lower central IS costs in the short term, it does not help the enterprise lower total overhead costs as rightsizing does. More to the point, an enterprise that outsources its IS resources is now faced with the very real probability that it will not be able to use IS technology to address critical operational issues due to the costs associated with modifying or defaulting on the outsourcing contract.

Audience Notes:

IBM

Rightsizing Product Strategy

DB2 on ES/9000 -- DB2 on RS/6000 -- DB2 on OS/2

All systems networked using DCE

Significant Issues For Interex Members

RS/6000 product transition imminent -- processor (RIOS to PowerPC) and compilers to be changed

Marketing messages are becoming fanatical as IBM reduces workforce dramatically

AberdeenGroup

Aberdeen Notes:
IBM is taking a very different approach to what it calls "open, client-server" systems than HP. At IBM's last major corporate briefing, IBM described its rightsizing vision as one of DB2 running on the mainframe, midrange, and PC with all systems communicating through DCE.

IBM is taking the RS/6000 through yet another technology transition to attempt to make it right. IBM has announced that by 1995 all its major platforms (ES/9000, AS/400, RS/6000, and PS/2) will be based on the PowerPC processor jointly developed with Motorola. This is a major technical challenge for a company that has had so few successes recently. And public perceptions may be clouded by an IBM staff that is now willing to say just about anything to avoid being part of the 28% personnel downsizing effort underway.

Audience Notes:

Sun Microsystems

Rightsizing Product Strategy

Multiprocessing SPARCserver/SPARCcenter servers interoperating with Solaris on client

Two-tier client-server architecture is sufficient

Significant Issues For Interex Members

Sun is still going through Solaris 1.X to Solaris 2.X transition glitches

Assumes commercial middle manager users will accept Unix on the desktop

AberdeenGroup

Aberdeen Notes:

Sun Microsystems is aggressively promoting its two-tier client-server architecture consisting of Unix on the client-desktop with upgradeable, multiprocessing power built into the server. The glue to make the client and server work together is primarily software provided by independent RDBMS and application suppliers.

Users continue to report that a considerable amount of technical expertise is required to support a Sun system as Sun is technically struggling with the transition between generations of operating systems and hardware platforms. In addition, the majority of end-users want an MS-DOS/Windows client on the desktop — a SPARCstation has too many drawbacks (price, source availability, mobility) to be acceptable in 1993.

Audience Notes:

DEC

Rightsizing Product Strategy

64-bit Alpha with either OpenVMS, NT, or OSF/1 on all computing systems within the enterprise

Production operating environments are in flux

Significant Issues For Interex Members

64-bit Alpha applications will not be available until mid-1994

DEC is attuned mostly to the needs of its installed base during current headcount reduction phase

AberdeenGroup

Aberdeen Notes:
DEC has bet the company's ability to capture new accounts in the mid-1990s on the success of the 64-bit Alpha processor and OpenVMS, NT, and OSF/1. However, while these may be the underlying technology components, Digital has a considerable amount of work to complete before its operating environments, including languages and networking, are compatible.

Alpha's strength is its design for fast integer processing – its weakness is a lack of volume sales to justify expanded R&D spending to ensure long-term scalability. While Alpha/OpenVMS will help meet the performance and price/performance needs of DEC's installed base, there seems to be little general interest on the part of vendor-neutral IS executives today.

Audience Notes:

Microsoft

Rightsizing Product Strategy

NT on Intel and Risc processors for server applications in 1994 -- replaces OS/2

Windows to Chicago to Cairo on desktop -- and then Cairo on server in 1995

Significant Issues For Interex Members

Windows/DOS is what users want on desktop

NT in 1993 could be a CLD

AberdeenGroup

Aberdeen Notes:
While Microsoft dominates the desktop in the world of rightsizing, it still is seeking a position offering the server software-of-choice. Its latest entry in server software is Windows/NT. The questions many IS executives are facing are whether the product is stable enough for production applications and will Microsoft support it long term.

Based on initial evaluations, Aberdeen Group is advising its clients to be very cautious regarding investments in Windows/NT in 1993.

Audience Notes:

Hewlett-Packard

Rightsizing Product Strategy

PA-RISC powering better than mainframe servers

Better than mainframe functionality in three-tier plus topology through leading systems software

Significant Issues For Interex Members

PA-RISC will need improved integer performance to stay ahead in 1994

HP is finally advertising to CEO decision makers

AberdeenGroup

Aberdeen Notes:
Hewlett-Packard has solved *all* the technical problems, especially I/O bandwidth, security, performance, and backup, that have in the past restricted the use of midrange systems to the departmental level.

The product reasons HP systems continue to be the rightsizing platform-of-choice is the robustness of the systems software (middleware), scalability and power of PA-RISC, and extensibility of the entire architecture.

Now that HP is finally advertising directly to the CEO level within enterprises, IS executives responsible for HP installations should anticipate more requests for information about how HP-based computing is both different and better than traditional IBM mainframe methods.

Audience Notes:

HP-experienced IS Role

- **Proactively establish senior management vision of how enterprise can win through IS rightsizing**
- **Enthusiastically describe today's multiuser technology capabilities to mainframe IS and line managers**
- **Base IS decisions on ROI to total enterprise -- not just IS**
- **Be a catalyst to create culture of change as a way of life within IS**

AberdeenGroup

Aberdeen Notes:
Just because Interex members understand how rightsized architectures operate, you cannot assume that either senior line management or your mainframe-trained IS peers do.

Senior management is looking for you to proactively propose new ways to run your enterprise's operations. Non-technical managers do not know the questions to ask to start or promote the rightsizing process — it is now your obligation to assume more responsibility for the business benefits of IS advances. This necessitates the ability to base IS decisions on the financial benefits to the entire enterprise as well as technical capabilities.

Audience Notes:

Rightsizing For Success

IBM, Unisys, NCR, DEC, SUN, ICL, Olivetti, DG, SNI, Bull, and others tell Aberdeen that HP is the industry leader for Rightsizing multiuser platforms

The obvious conclusion is that:

Interex members are the most qualified in the world to lead their enterprises by rightsizing for success

AberdeenGroup

Aberdeen Notes:
Interex members now face the challenge of managing success within their enterprises. They now find themselves in charge of the right platform and with the right experience. Others IS professionals using supplier technology that did not stay ahead are racing to catch up to you.

But just being technically qualified is not enough — you must be individually motivated to use this technical knowledge to successfully direct your enterprise's transition to adopt the rightsizing approach to IS. The management concepts on which ACTION are based can provide you with a foundation for making the correct rightsizing technical and business choices both now and in the future.

Audience Notes:

Paper 1000

Using RTE More Effectively in an Increasingly non-RTE HP World

F. Stephen Gauss
U.S. Naval Observatory
3450 Massachusetts Ave. NW
Washington, DC 20392-5420
202-653-1510
fsg@sicon.usno.navy.mil

In recent years RTE has been evolving in ways that make it easier to use in a multi-computer environment, especially when the other computers are based on the Unix operating system. In many cases the RTE implementation is actually friendlier than the HP-UX implementation, but the important point is that the computer user should be able to move from system to system with relative ease. With improvements made to the networking interface in recent releases, it is quite practical to use workstations to open multiple windows into the HP1000 and to have many of the advantages of a windowed environment even though RTE does not formally support such a thing. Recent products released by both the RTE and HP-UX support groups have provided many of the tools to ease the use of RTE in the non-RTE environment, especially if the other systems are HP-UX. The Gfox product provides a graphical terminal interface that is faster than the terminal it is emulating. Softbench/1000 permits the use of the HP-UX program development tools with RTE, while at the same time some of the HP-UX tools have been made available directly in RTE. And with an Edit/1000 version available from a third party for HP-UX, the user can work in whatever environment he is most comfortable with and can use the tools with which he is most familiar.

However, there remain a number of areas where integrated solutions do not yet exist. Methods have been devised to work around some of these areas and they will be described in the remainder of this paper. Some involve features provided at release 6.0 of RTE, some interact with HP-UX systems via Berkeley Sockets and some provide temporary workarounds for features that are known to be under development at HP. In cases where code can be modularly developed to support the operation, that code has been contributed to either the 1993 CSL or the 1993 swap tape. Where modules are required on both the RTE and HP-UX platforms, those modules have been contributed to both CSL's.

E-Mail Support via Modem

In the Unix world a system of inter-machine communication called uucp was developed to allow computers to talk to each other regardless of the mechanism available. RTE has a fine E-mail support system, consisting of Mail/1000 and either NS/DS1000 or the ARPA smtp service. However, one of our RTE systems is located in another country and, for a number of reasons, no network connection is available. There are dial-up modems operating at 9600 baud and a scheme has been implemented to allow standard E-mail to be passed back and forth between RTE systems in batches. This process has been given the name "bulk-mail". The CONNECT program from ICT is used to provide the communication interface. Because of the delays in the signal when communicating half way around the world, the ENQ-ACK handshaking used on HP computers is unsuitable. The ports and the modem must be set to use XON-XOFF. Because certain RTE programs use some of the handshaking characters for identification purposes, it is important that all of the XON-XOFF characters be passed through the modems. Most modems have such a "pass-thru" mode associated with the XON-XOFF menu choice. In addition the HP1000 mux port must be set to half-HP mode (203b), which allows both XON-XOFF and ENQ-ACK to appear on the port, even though only XON-XOFF is used for handshaking.

Bulk-mail transfers can be initiated from either end of the connection. For purposes of illustration, let the computer where the operator is located be called the local system and the unattended computer at the other end be called the remote system. At release 6.0 a feature was added to Mail/1000 allowing mail directed to a specific node to be diverted into a directory. The node name of the remote computer is given the same address as the local computer. Thus, any mail sent to the remote computer's node name arrives at the local computer and is redirected into a directory called /MAIL/BULK_OUTMAIL. Since the remote computer is NOT connected to the same network, there is no problem in assigning the two node names to a single address. At this point the remote computer appears to be a perfectly legitimate E-mail address. However, all mail sent to it ends up as files in the above mentioned directory on the local computer. Note, though, that within those E-mail files, the To: address specifies the name of the remote computer, not the local computer. The next step is to make a modem connection using CONNECT between the two systems and to use Kermit to transfer all the files in the

/MAIL/BULK_OUTMAIL directory to a /MAIL/BULK_INMAIL directory on the remote computer. In the same operation all files in the /MAIL/BULK_OUTMAIL directory on the remote computer are transferred to the /MAIL/BULK_INMAIL directory on the local computer. Thus, all accumulated mail on both systems is transferred to the other system. This can be done at a scheduled time (when phone rates are low, for example) or whenever someone dials up the other system for any other reason.

Once the files are transferred, the sendmail program is scheduled on each of the files. Since the node name referenced in each message is now the real node name of the computer that the mail is on, the mail is delivered to the proper address.

There are a few caveats; mail sent to lists of names on the remote computer is delivered to each name at the local computer, but then when the file is transferred to the remote computer it is redelivered to the same list, thus causing multiple messages to be delivered. This can be avoided by editing the file to use the TO: line rather than the To: line. Occasionally, a transmission problem will cause the process to abort, so as the mail is sent off it is also copied into a /MAIL/BULK_SCRATCH directory, which can be purged every once in awhile.

This system works reasonably well and provides a transparent method for passing standard E-mail over modems. It requires several transfer files for CONNECT and EDIT/1000 and they are shown in Appendix A. These files are provided here for purposes of illustration only. While they are slightly modified copies of working files, no guarantee that they will work as shown is implied.

Remote HP-UX Services

When working in a mix of RTE and HP-UX environments it is convenient to have some frequently used information available on both systems. An efficient way to pass information back and forth is via Berkeley Sockets. Once a client-server relationship has been established, it is quite feasible to send almost any kind of information between the two systems. Such a mechanism has been presented previously in both a paper ("Using BSD IPC to Synchronize RTE and HP-UX System Times", King, Proceedings of the Interex 1992 Conference) and programs contributed to the CSLs (King, Schmidt and Gauss, 1992 CSL/RTE and CSL/HP-UX). Wendy King and Rich Schmidt developed client-server programs to synchronize time between an HP-UX system and an RTE system. Since either one could act as the source of the time, four modules were developed to allow either system to be the client and either one the server. Using the same basic code, it was relatively easy to create a new set of programs to provide information about the users logged on to either system. The program WHOZN had been written for RTE and allowed an RTE system to show who was logged onto itself or any other RTE system. It used both NetIPC and DS transparency so that it worked with both RTE-A and RTE-6. It has since been enhanced to add a BSD module, which permits it to access HP-UX systems as well. In addition the complementary modules were written for HP-UX to allow the same functions to be performed from that end. Thus, a user on an HP-UX system can display users logged onto that system, other HP-UX or RTE systems and a user logged on to an RTE system can see who is logged on to other RTE or HP-UX systems. The user runs the client which connects to a server module on the other computer and supplies the required information.

However, WHOZN was somewhat complicated to write, since it needs to access system tables in RTE to obtain the necessary information. A new set of programs (r_server, r_client) was written to handle the simpler situation where all that is necessary is to have the remote computer run a program to obtain the information and then pass it back to the local (calling) computer. This program pair provides an NSLOOKUP function for RTE, displays disk space information for both RTE and HP-UX systems, allows HP-UX manual pages to be displayed on the RTE system and allows a method of communicating with a user on another system.

The programs take advantage of a new feature at RTE release 6.0, symbolic links, that provides a way to create multiple names for the same file (or program). Using symbolic links, a single program can be called by several names. The program can examine its runstring to determine which name was used to call it and act accordingly. Thus, the single program, r_client, can be called by the names nslookup, bdf, man and to_user.

In each case the sequence of events is as follows:

the r_client program
  - o determines which service is to be provided
  - o sends the service number and any parameters to the other system
  - o sends any data to the other system
  - o waits for a completion

the r_server program

- is scheduled by inetd
- retrieves the service number and parameters from the input stream
- schedules a program or shell script, passing it the necessary parameters and directing its output into a temporary file
- reads the temporary file and sends it back to the calling system

The operation of the program when called by each of these names is as follows:

nslookup, node name, [remote system]

nslookup is an HP-UX utility that searches the network to determine the IP address of a given node name. This is an easy way to determine whether you have the right spelling for an address or whether such an address exists. If nslookup doesn't know about it, there is no point in sending E-mail to it! The RTE user types nslookup and the name of a node that is to be looked up. The name of a system where the nslookup utility is to be run can, optionally, be given, but the program provides for a name to be built in as a default. This system should be a name server or have access to a name server and must be running the r_server part of the pair. The node name is passed to the remote system, which runs nslookup on the node name and returns the information that it finds to the calling program which displays it.

bdf[, remote system]

bdf is an HP-UX utility that displays the free space available on the disks. This can be useful to know before transferring large amounts of data onto the system. The name of the remote system whose disk space is desired must be given. If no system name is given, then the local system is assumed. As in the previous example, the remote system runs the bdf utility (or, if it is an RTE system, FREES +M) and passes the information back to the calling system.

man, man page[,remote system]

man is an HP-UX utility that displays the manual pages for a given subject or utility. Its format is not particularly friendly. This function only makes sense when run from an RTE system, since all HP-UX systems have the same man pages. In this case, the request is sent to the HP-UX system which then opens an anonymous ftp connection back to the RTE system (for information on setting up an anonymous ftp account, see below). It then transfers the man page to the RTE system. The man program (r_client) on the RTE system then lists ways to display the man page using the more friendly RTE utilities. The information is given in a way that the command stack can be used to access them. Both Edit/1000 and LI can be used, although, since man pages tend to be full of HP-UX control sequences (tabs, etc.), it is sometimes faster to use Edit/1000 and kill the HP-UX tabs.

to_user, logon name[, remote system]

to_user is used to send a message from an HP-UX or an RTE system to a user logged onto the same or another system. The logon name must be the name that the user is logged on by. If no remote system is specified, the message is sent to the local system. To_user is interactive and allows you to enter a message. At the conclusion of the message it is sent to the user. If the user is located on an HP-UX system, the utility xmessage is used to put the message into an X-window. Note that the xmessage utility is a previous CSL contribution that creates a message window on the user's session. If the user is located on an RTE system, the message is simply copied to a temporary file and the program BC is used to inform the user of this fact. BC is a previous CSL contribution that puts a single line message in the soft key area of the screen without disrupting the user's display. If the user is not logged on, an appropriate message is returned. The user name "all" can be used to send the message to everyone who is logged on.

The services provided by r_server and r_client can be easily expanded by simply adding modules to the program. The connection across the network is provided automatically by one-line entries in the /etc/inetd.conf and /etc/services files, which now exist on RTE as well as HP-UX systems. A file, /etc/r_server.file, contains entries for the name of the local system and the name of the default remote system.

Anonymous ftp

HP-UX systems generally have an anonymous ftp account, which allows any user to connect to a directory for the purpose of uploading or downloading files. Many system managers prefer to have files required by outside (anonymous) users placed in this temporary directory, rather than directly into user accounts. It provides better security, since the user does not need to know a password, has no access to user directories and need not even know about the accounting structure on the system. Furthermore, the directory can be managed such that large file sizes will not use up critical disk space. This type of account can be set up on an RTE system in the following way. An anonymous account with no password is created using GRUMP. To prevent anyone from logging on to this unprotected account, the startup command (normally RU,CI) should be an invalid string (I use RS). The home directory for the account should be placed on a disk LU by itself or in an area where no harm will be done if a user fills the directory with a very large file. The account should have as many LU's disabled in the LU map table as possible. This account is also useful for programs that must automatically transfer data to the RTE system, since it does not require that a password be written into a program. The previously described man utility makes use of this account to transfer the man pages from the HP-UX system to the RTE system.

The addition of Berkeley Sockets along with HP-UX-like utilities to RTE has provided considerable flexibility in developing interactions between the two. It is becoming increasingly easy to move between RTE and HP-UX, which has enhanced the attractiveness of adding HP-UX systems to an RTE network, while extending the useful life of the RTE systems.

Appendix A

Command files that make up the bulk_mail operation.

File 1. /cmdfiles/bulk_mail.cmd
Command file on remote system to initiate the transfer of mail between the local and the remote systems

```
set direct = $wd
* Set up inmail directory to receive files from remote
* Set up outmail directory to send files to remote
*
wd /mail/bulk_inmail
pu   /mail/bulk_outmail/tempfile.tmp
edit -b /mail/bulk_outmail/tempfile.tmp i&&&¦er
pu @.tmp ok
*
*  Start kermit in server mode
*  Break back to CONNECT and run the command file
*  on the inmail directory
*
echo '========================================'
echo '== Enter your CONNECT break character =='
echo '== then tr,bulk_mail                 =='
echo '========================================'
echo ' '
kermit server
*
* All files transmitted in both directions
* Prepare a command file that will pass the
* file names to sendmail
*
pu send.cmd
pu tempfile.tmp
dl @.tmp s send.cmd
if is $return1 = -50 -i
then
  echo 'No files transferred TO.'
else
* Create the command file needed to pass the file names
* to sendmail and transfer to that command file
  edit -b send.cmd seasof¦tr,/mail/bulk/send.edit
  echo 'Sending mail to addressees.'
  tr,send.cmd
fi
*
* Preserve the files in a temporary directory in case
* of transmission problems
*
mo /mail/bulk_outmail/@.tmp /mail/bulk_scratch/
wd $direct
unset  direct
```

File 2. /connect/bulk_mail.cmd
Command file on local system to carry out transfers interacting with the remote system

```
*
*  This command file is stored on the directory
*  from which CONNECT is run and the modem is
*  operated (assumed to be /connect)
*
*  Switch the working directory to inmail
*
ru,ci,/mail/bulk_inmail/sd.cmd,/mail/bulk_inmail
pu /mail/bulk_outmail/tempfile.tmp
*
*  Create a command file for kermit to communicate with
*  the kermit server that is now active on the other
*  end
*
edit -b  /mail/bulk_outmail/tempfile.tmp i---¦er
*
*  Send the mail files
*
kermit tr /mail/bulk/send_mail.krmt
of,kermit
*
*  Escape back to connect and complete the operation
*  of mailing out the files that were sent to the
*  local end
*
` ru,ci,/mail/bulk/finish_get.cmd
` ru,ci,/mail/bulk_inmail/sd.cmd,/connect
x
&&&
```

File 3. /mail/bulk_inmail/sd.cmd
Command file for CI to switch working directories (required by CONNECT)

```
wd,$1
```

File 4. /mail/bulk/finish_get.cmd
Command file to complete the transfer and send the mail on the local system

```
*
*  Mail out all of the files that were passed from the
*  remote system to the local system
*
wd /mail/bulk_inmail
*
* Save files in case of transmisison problems
*
mo /mail/bulk_outmail/tmsg@.tmp /mail/bulk_scratch/
pu send.cmd
pu tempfile.tmp
*
*  List all received files into a file that will be
*  edited into a command file
*
dl @.tmp s send.cmd
if is $return1 = -50 -i
then
  echo ‘No files transfered FROM.‘
else
*
* Create a command file that passes the names of the
* mail files to sendmail then transfer to that file
*
  edit -b send.cmd seasof¦tr,/mail/bulk/send.edit
  echo ‘Sending mail to addressees.‘
tr,send.cmd
fi
ex
```

File 5. /mail/bulk/send.edit
Edit a directory list file into a sendmail file

```
$
p&&%%%&&&
1
1,3,d,/&&&%%%&&&/,a
sele20
bk
sele
sewc1,1
1,$,u//tr,\/mail\/bulk\/sendout /q
sewc
1,$,d,/&&&%%%&&&/,a,v
er
```

File 6. /mail/bulk/sendmail.krmt
Kermit command file to communicate with the kermit server

```
send /mail/bulk_outmail/@.tmp
get  /mail/bulk_outmail/@.tmp
finish
quit
```

File 7. /mail/bulk/sendout.cmd
Sends out the mail on the remote system

```
*
*  For each file, change any occurrences of
*  "apparently to" to "to", make sure that the correct
*  node is used in the case of LHOPs and then pass the
*  file name to sendmail
*
set file = $1
edit $file seasof¦`1,$,x/Apparently-To:/To:/¦er`
edit $file seasof¦tr,/mail/bulk/node.edit
sendmail -s $file
```

File 8. /mail/bulk/node.edit
Edit file to ensure that the proper "To" routing is used

```
sereon
secfof
f/TO +\</
x/TO /TO\:/
secfon
.+1 $ x/To\:/To /
b/to\: @\@alpha/
x/{to\: @}\@alpha.@/&1
x/{@}%{.[^%]}/&1@&2/
x/\<//
x/\>//
er
```

Paper 1001

Transferring CPLOT and LPLOT Graphics from the HP 1000 to a PC

by Gerald Lisowski and Dan Kukla
ZENECA, Inc.
Western Research Center
1200 S. 47th Street
Richmond, California 94804
510-231-1390

In the PC world HPGL has become a *de facto* graphics standard. Most graphics plotters accept HPGL input and most graphics programs can produce HPGL output. Recently high end word processors, such as WordPerfect by WordPerfect Corp. and Word by Microsoft, have acquired the ability to import HPGL graphics. They can import the graphics either directly (Microsoft Word) or through a conversion program (WordPerfect).

Incorporating graphics into our documents is an important feature. Before our word processors obtained this ability, the only work around was to create the graphic separately and leave space for it in the document. If the graphic occupied space in the body of the text we would usually paste the graphic and document together and photocopy the resulting page. This increased the time and effort needed to produce a document. If something was later added or removed, it usually meant the whole process would need to be repeated, typically with a lot of aggravation. If the graphic occupied a whole page, things were a little bit easier. But even with full page graphics we often need to add captions, running heads, or page numbers. We still had to go through the paste and photocopy routine. Now we let the word processor handle the layout details. It does it faster and smoother than we ever could.

One type of graphic we are especially interested in is chromatograms. When a chemist injects a mixture of compounds into a machine called a chromatograph, the chromatograph separates the compounds and sends them to a detector. The detector sends the output to a device capable of accepting its signal. If the device is a recorder a chromatogram is obtained. A chromatogram plots detector response on the y-axis verses time on the x-axis. The chromatogram contains two valuable pieces of information, the retention time and peak area. The retention time is how long the compound spent in the chromatograph, and can help the chemist identify the compound. The peak area is proportional to how much of the compound was placed in the chromatograph. Using this information the chemist can quantitate how much of the material was in the original sample. Figure 1, supplied by HP as part of its Laboratory Automation System (LAS) training course, is an example of a chromatogram. Figure 2 is the same chromatogram with the base lines drawn in after processing by LAS.

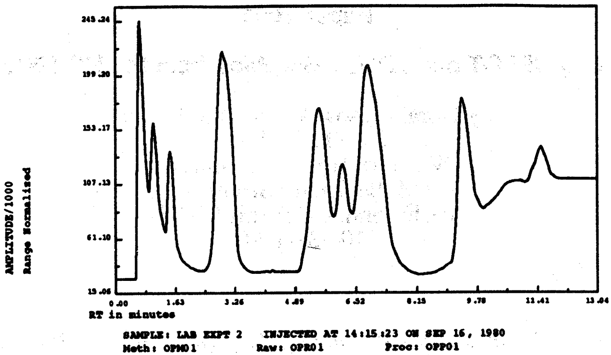


Figure 1. Raw chromatogram

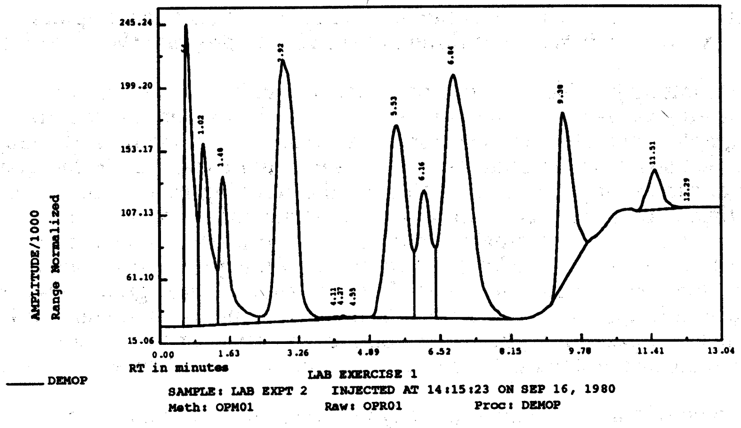


Figure 2. Processed chromatogram

A recorder is not the only device capable of accepting the signal from a chromatographic detector. The signal also could go to a computer with the hardware necessary to receive the signal, and the software necessary to process it. Such a system is called a chromatography data system. Given that HP is a leader in analytical instrumentation and computers, it's not surprising that they make a chromatography data system. In fact they make several. One family of these, the LAS systems, are based on the HP 1000. The system used on an E/F series computer running RTE-6 is the 3357 while the system used on an A series computer running RTE-A is the 3350A. Both units can store the raw data points to allow for subsequent reanalysis and replotting of the data.

One program that can replot the data stored in a 3357 or 3350A is CPLOT. Another program, available only for the HP-3350A is LPLOT.[1] Both programs are available from Hewlett-Packard.[2] CPLOT and LPLOT use Graphics/1000-II and can take the chromatographic data and replot it on a graphics terminal, or send the plot to various graphics printers or plotters. Two supported devices are the HP7475 and HP7550, both HPGL plotters. CPLOT output also can be sent to a spool file. Using CPLOT we can send the output to a spool file that CPLOT thinks is an HP7475 or HP7550. LPLOT is able to directly output to a file. The file is a text file that can be sent to a PC using a variety of programs. The program we use is Reflection 7 by Walker Richer & Quinn, Inc.

Preliminary setup

CPLOT

On both RTE-6 and RTE-A systems, file "USERC or /SYSTEM/USERC.TXT needs to be set up correctly before CPLOT can be used to produce an HPGL file. It must contain spool table entries for at least one of the two supported HPGL plotters. The text below was taken from a "USERC file set up to allow the spooler to emulate either the HP7475 or the HP7550.

```
!SPOOL
 spool table - line 1=no. of entries, each entry on separate line:
 wsp name, plot limits x1,x2,y1,y2,direction code,aspect ratio
==>2
W7475,0.,0.,0.,0.,250,.6232
W7550,0.,0.,0.,0.,250,.7700
```

Information on what the entries mean and how to modify "USERC or /SYSTEM/USERC.TXT can be found in the CPLOT manual.

You also may need to modify the pen section of "USERC or /SYSTEM/USERC.TXT to have all the pens be 1. When Microsoft Word prints an incorporated HPGL file, the characters drawn with pens other than 1 are fuzzy. The characters drawn with pen 1 are nice and sharp.

If CPLOT is run from CI, the work station programs (wsp files) must be in the PROGRAMS or working directory.

LPLOT

For LPLOT you need to have a device definition file. These are usually in the /LAS/DEVICE directory and have the extent DEV. These files are described in the LPLOT manual. Although not necessary, a PLOTTOFILE.DEV file may be used. If this file does not already exist it can be prepared by copying an existing HP7550 device definition file. The files are text files and can be modified with EDIT. The following lines may need to be changed.

LU = *Any valid spoolable LU.*

Workstation = */programs/w7475.run or /programs/w7550.run*

XAspect =
YAspect =

These should be set for the aspect ratio of the surface to be printed on. For example on a 6 inch wide piece of paper, Microsoft Word for DOS says a 4.411" high picture will preserve the aspect ratio of the picture. For the best fit for these circumstances, the XAspect would be set to 600.0 and the YAspect would be set to 441.1.

Producing the HPGL file on the 1000

The first step in getting the PC graphic is to create the HPGL file on the 1000.

CPLOT

CPLOT does not have a procedure to directly plot to file. Two indirect procedures can be used, interactive and batch.

Interactive mode

The interactive mode is available only with the RTE-6 version of CPLOT. It's more involved than the batch mode, but also more flexible.

The first thing is to set up the spool file that will contain the HPGL commands. Log on with a capability of at least 30. Next use the CR command to create a type 3 or 4 file to hold the HPGL commands. With the CR command you must specify a size for the file. A size of 24 is suggested. This is the default size for scratch files, and the *Programmer's Reference Manual* recommends it for user created files. If more file space is needed the system will automatically create it. Next issue the command RP,SMP, if SMP has not already been RP'd. Finally issue the command SL,lu,filenamr,WR. LU must be in your SST and should not be in use. Filenamr is the name of the type 3 or 4 file created above with the CR command. You now are set up to redirect output from an lu into a file.

Next run CPLOT. In CPLOT select your raw file and any other parameters you wish to modify. If necessary, you can modify, plot, and analyze until satisfied with the results. When all the parameters are set press F4, the menu function key. Then press F5, the plotter function key. In the plotter menu, press F6, and type the number of the spool LU. Then press F7 to choose the plotter. Type W7475 or W7550, whichever is in the spool table in "USERC. Plot to the spool file by pressing the plot function key, F5. After plotting is complete press F4, the menu function key. Finally press the end key, F8, to exit, and answer yes to the "*ARE YOU SURE YOU WANT TO QUIT*" question. At this point you will be in graphics mode. If you are using Reflection, get back to alpha mode by pressing alt-7 on the numeric key pad.

Once you are back into RTE type CS,lu,EN to close the spool file. If you do not want to keep SMP around, and you have the capability, type OF,SMP.

Batch mode

Batch mode can be used with the RTE-6 version of CPLOT, and is the only way to get an HPGL file from the RTE-A version. As in the interactive mode, the RTE-6 version of CPLOT requires setting up a spool file to contain the HPGL commands. This is done using the same commands described above for the interactive mode.

Spooling HPGL output on older version RTE-A systems requires that the appropriate plotting device is present in the system, and its LU is below 64. A dummy LU can be generated for plotting purposes. The following is an example of an Interface and Device Table entry for an RTE-A answer file to generate a dummy LU (LU 9) to support plotting to HP7475 and HP7550 plotters:

```
IFT,%ID*37::RTE_A,SC:27B
DVT,,,TO:2000,DT:77B,TX:0,DX:1:36B,PR:0,LU:009
```

Newer versions of RTE-A do not require this dummy LU, or that the plotting device be present in the system. When the output LU is specified, an unused LU should be designated. This will insure that legitimate output is not accidentally routed to a spool file.

Turn spooling on to redirect the output from the LU to a file by entering SP,ON,LU,FILENAME, where LU is the number of the plotter LU and FILENAME is the name of the HPGL file you want to create.

After the LU is set up to direct its output to a spool file, generate the plot by running CPLOT in batch mode. On an RTE-A system the spooling (SP) option must be used along with the codes for adding retention times, names, and base lines to the plot. This is done by issuing the command CPLOT,WSP,LU,,RESULTFILE,,SP... at the system prompt. The run string parameters are described in the CPLOT manual.

As an example, on an RTE-A system the commands

```
SP,ON,9,PLOT.PLT
CPLOT,W7475,9,,DATA.RES,,SPBL
```

will redirect output from LU 9 to the file PLOT.PLT and generate a plot emulating an HP7475 based on the chromatographic data in the file DATA.RES. The plot will have base lines drawn in.

On an RTE-6 system the spooling option must not be used. For an RTE-6 system the equivalent set of commands is

```
CR,PLOT::-17:3:24
RP,SMP (if SMP is not already RP'd)
SL,33,PLOT::-17,WR
CPLOT,W7475,33,,DATA,,BL
```

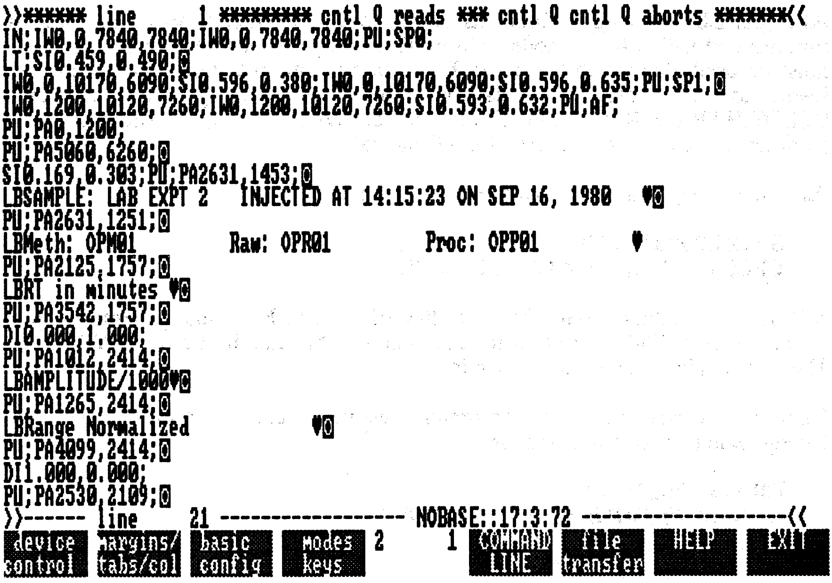
On an RTE-6 system the spool file should be closed after the plotting is finished. As in the interactive mode type CS,lu,EN. On an RTE-A system the spool file is automatically closed when the plotting is completed.

LPLOT

LPLOT has the capability of creating an HPGL file as a menu choice from within the program. First get the plot on the screen and set any normalizations. The output will be determined by the specified normalizations, not what is on the screen. From the top menu select OUTPUT. Then choose **Select Device** from the side menu. The program will ask for a device definition file name. Type in the appropriate file name, for example PLOTTOFILE. Finally select **Output to File** from the side menu. The program will ask for a plot file name. After the name is entered LPLOT will send the plot to the designated file.

Transferring the HPGL file to a PC

You now have the HPGL file on the 1000. How do you get it to the PC? If the HPGL file was a normal text file the answer would be simple. Do a LOG BOTTOM with the TO DEVICE set to TO DISK. Then copy, store, or dump the file to LU 1. Unfortunately this won't work with HPGL files. HPGL files are not normal text files. To see why, run EDIT on the HPGL file. If you list it out it looks like a regular text file. However if you go into screen edit mode you get something like figure 3.



```
))****** line      1 ********** cntl Q reads *** cntl Q cntl Q aborts *******((
IN;IW0,0,7840,7840;IW0,0,7840,7840;PU;SP0;
LT;SI0.459,0.490;▣
IW0,0,10170,6090;SI0.596,0.380;IW0,0,10170,6090;SI0.596,0.635;PU;SP1;▣
IW0,1200,10120,7260;IW0,1200,10120,7260;SI0.593,0.632;PU;AF;
PU;PA0,1200;
PU;PA5060,6260;▣
SI0.169,0.303;PU;PA2631,1453;▣
LBSAMPLE: LAB EXPT 2   INJECTED AT 14:15:23 ON SEP 16, 1980   ♥▣
PU;PA2631,1251;▣
LBMeth: OPM01          Raw: OPR01          Proc: OPP01          ♥
PU;PA2125,1757;▣
LBRT in minutes ♥▣
PU;PA3542,1757;▣
DI0.000,1.000;
PU;PA1012,2414;▣
LBAMPLITUDE/1000♥▣
PU;PA1265,2414;▣
LBRange Normalized          ♥▣
PU;PA4099,2414;▣
DI1.000,0.000;
PU;PA2530,2109;▣
))------ line     21 ------------------ NOBASE::17:3:72 -------------------((
```

Figure 3. EDIT screen of an HPGL file.

The funny looking characters[3] at the end of some of the lines are control characters that are an integral part of the HPGL file. In Reflection a simple LOG BOTTOM strips out these necessary characters. A way must be found to transmit these characters to the PC. In Reflection 7 this is done with the CAPTURE command.

The CAPTURE command passes through all characters. The following command file can be used to transmit an HPGL file to a PC.

```
DISPLAY "What is the host filename?"
ACCEPT V1
DISPLAY "^M^J"
DISPLAY "What is the PC filename?"
ACCEPT V2
DISPLAY "^M^J"
TRANSMIT "**,$1,1"
OPEN $2
SET CAPTURE YES
TRANSMIT "^M"
WAIT FOR "^Q"
CLOSE DISK
SET CAPTURE NO
```

Where ** is DU for an RTE-6 system and CO for an RTE-A system. The HPGL file is now ready to be used by MS-DOS programs.

Using the above command file causes some header and ending "junk" to be introduced into the PC file. This seems to present no problems, either to programs that use the file directly, such as Microsoft Word, or programs that translate the file into their own format, such as Word Perfect.

One fact needs to be mentioned about the HPGL file. It can contain a lot of space around its border. CPLOT output to an HP-7550 using the settings described earlier produces the least amount of extraneous space. Figure 4 is a directly imported HPGL file with a box around its perimeter. It is the file used to produce figure 1.

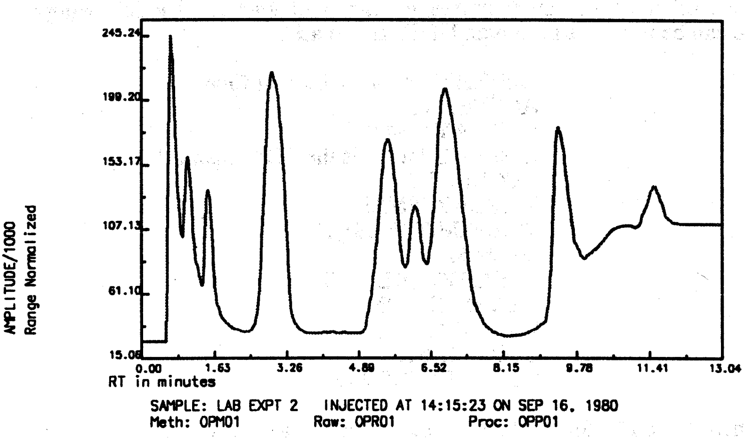


Figure 4. Directly imported HPGL chromatogram

You can see that there is a lot of blank space around it, especially at the top and bottom. If you use a program like WordPerfect for Windows, with its own graphics editing, this is only a minor annoyance. On the other hand if you use a program like Microsoft Word, with no graphics editing capabilities, you probably will want to use another program to edit out the superfluous blank space.

Although we have described using CPLOT and LPLOT to generate the HPGL file and Reflection 7 to get the file to a PC, the techniques we have demonstrated can be adapted to other programs. Any HP 1000 program that can send graphics to an HPGL plotter should be able to have its output redirected to a spool file, either directly or indirectly. Any good terminal emulator should allow you to copy the HPGL file from the HP 1000 to your PC. Just remember to make sure that it passes control characters as well as normal alphanumeric characters.

References

1. HP19471A Advanced Graphic Chromatogram Processor.

2. Hewlett-Packard sells and supports CPLOT for the HP-3350A. CPLOT is bundled with new HP-3350A systems. HP no longer sells CPLOT for the HP-3357. If you wish to obtain CPLOT for an HP-3357 contact your sales rep.

3. Figure 3 was taken from the screen of a PC with a Hercules compatible graphics card. The control characters are shown in their IBM character set representation. HP terminals display the control characters differently.

Golf and the HP1000

Dave Medlicott

Hewlett-Packard
Software Technology Division
11000 Wolfe Road
Cupertino, CA.

Introduction

For the past several years, one of the projects that everyone in the RTE lab agrees we must do, but which always seems to get put on the back burner until the last possible minute, is a good demonstration for the INTEREX conference. When we speak of a "good demo", there are a few key requirements that must be met: it must demonstrate an important capability of the HP1000, such as realtime response or network connectivity, it must be interesting (to at least a few people) and we must be able to do it. Thus, we have had to scrap such ideas as a mock factory floor and (one of my favorites) an automated chicken farm (where *real* chickens would lay eggs that would roll onto a conveyor belt, etc.).

This year, one of the proposals involved combining the favorite activity of some of the RTE lab engineers, golf, with work. The idea was a realtime golf swing analyzer. When a golfer swings the club at a golf ball, many factors affect how the ball will travel after contact has been made. These factors include the weather, the ball construction, the club construction and the various components of the swing itself. At impact, the major components of the swing that will affect the ball's flight path are: the speed at which the head of the club is moving, the path that the club is moving along, the angle at which the clubface meets the ball and the location on the clubface where contact with the ball is made.

Our goal was to develop a device to measure these factors using the HP1000 as the realtime data collector and analyzer. Devices which measure these components and analyze the golfers swing are already widely available. However, these devices require dedicating a processor to the task. In addition to collecting and analyzing the data, our goal was to show that the HP1000 can do this while simultaneously performing other tasks. In other words, it is not necessary to dedicate your HP1000 to a single realtime task.

The three components of such a collection and analysis system are: the detector, the collector and the analyzer. The detector is a specialized device consisting of an array of photovoltaic sensors that detect a break in a light source. The output of these sensors is connected to an HP 12006A Parallel Interface Card (PIC) which is controlled by a privileged interface driver (ID55) on the HP1000. Data analysis is performed by a FORTRAN program on the HP1000. This paper will discuss the driver that was written for this purpose with an emphasis on why a privileged driver was required and the major differences between standard non-privileged drivers and privileged drivers.

Introduction to Drivers

In general, programs do not directly communicate with interface cards. The usual procedure is for a program to make an EXEC request to RTE, passing it the parameters specifying an input or output operation, buffer location and size and to which device the request should be directed. RTE then looks through its tables to determine which of the operating system modules will be called to perform the processing needed to execute the requested task. These operating system modules are the I/O drivers.

Under RTE-A, drivers may be classified as device drivers or interface drivers. This is because a variety of devices may be connected to the same interface card. The device driver packages device specific requests to the interface driver which then issues the I/O instructions that allow the interface card to perform the requested tasks. In many cases, including ours, it is possible that only one device will be connected to an interface card. In these cases, a device driver may not be required. This is the case for many devices connected to the PIC card. ID.50 is the standard PIC driver supplied with RTE-A. It is a non-privileged driver and usually there is no device driver associated with it. Though ID.50 would suffice for many of our tasks, there were a few requirements of our application which required a privileged driver. We will get to that in a few moments. First, however, we will examine the interface between the operating system and the drivers.

Operating System Functions

I/O Tables

The operating system must perform several tasks which ease the task of the driver designer. Most RTE supplied drivers are written to handle a wide variety of devices while most user written drivers are written for a very specific interfacing task. RTE implements a standard set of tables and protocols which allows drivers to be written with a standard set of procedures.

Among the tables which RTE maintains are the interface table (IFT) and the device table (DVT). The IFT contains an entry for every interface and the DVT contains an entry for every device. When an I/O request is made, RTE takes the information from the request and places it in the tables at locations known to the driver designer. This gives the driver the flexibility of servicing several interfaces or devices.

Driver Entry

Before entering the driver, RTE sets up the IFT and DVT for the current request. It may also be necessary for RTE to initialize an I/O port map. This is a map that describes the area of memory where the I/O buffer resides. When a user makes an I/O request with an EXEC call, the address of the I/O buffer is within the user's address space. If the I/O transfer is a direct memory address (DMA) transfer, then the transfer will begin executing and control may proceed on to a different user program. The map set which describes the new user program is different than the one which describes addresses used in the DMA transfer. For this reason, the memory map that is used for the DMA transfer must be independent of the memory map used for user program execution.

Prior to entering the driver, RTE sets a code in the A-register telling the driver why it is being entered. The codes and usages are as follows:

```
0   - Abort the request in progress
1   - Initiate a new request
2   - The interface has generated an interrupt (Continuation)
3   - The current request has timed out
4   - The system powerfailed
```

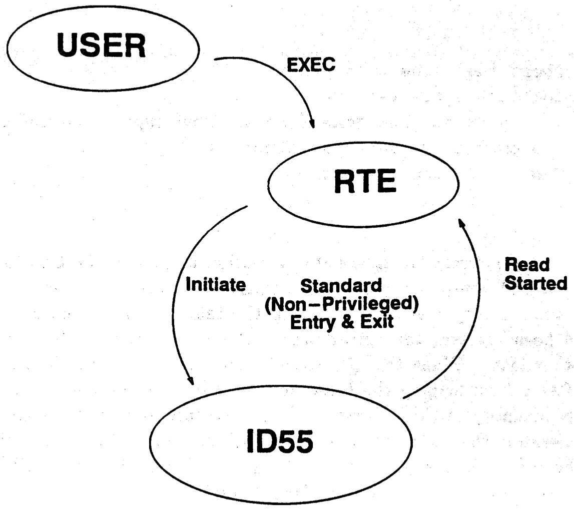
The entries of primary interest for us are the Initiation entry and the Continuation entry. For both privileged and non-privileged drivers, initiation requests are made through RTE. That is, RTE processes the request and sets up the tables as described above. Once the I/O transfer has been started, the driver may exit and return to RTE which then goes on to perform other tasks. When the I/O transfer completes, the interface card generates an interrupt. After acknowledging the interrupt, the next instruction to be executed will be the one at the memory location (trap cell) that corresponds to the select code of the interface card generating the interrupt. For all non-privileged interface cards this instruction is a branch to the central interrupt processor ($CIC) within RTE. For privileged interface cards, this is a branch directly into the interface driver.

For most non-privileged drivers, the sequence of operations is as follows:

```
1) An EXEC call is made to perform an I/O request.
2) RTE is entered and tables are initialized.
3) The driver is entered with an Initiation code.
4) The driver starts the I/O transfer and returns to RTE.
   with an indication that the I/O is underway.
5) The I/O transfer proceeds at the same time that RTE dispatches
   another user program.
6) The I/O transfer completes, generating an interrupt.
7) RTE is entered again ($CIC) which sets up the required tables.
   and passes control to the driver with a Continuation code.
8) The driver performs any clean up tasks and returns to RTE.
   with an indication that the I/O is done.
9) RTE  performs its clean up and returns status indicators to the
   program which made the EXEC call.
```

For privileged drivers, the sequence is the same up until step 6. Figure 1 shows the non-privileged entries and exits for our privileged driver.

**Standard (Non–Privileged)
ID55 Entries and Exits**



The Standard entry does the following:

- **Save IFT and DVT addresses needed later**
- **Configure DMA read transfer**
- **Enable STO interrupt**

Figure 1

Privileged drivers have two entry points. The first entry point is the same as for standard drivers except that there is no continuation entry. Continuation entries are made when an interface card generates an interrupt. For privileged drivers, the trap cell contains a branch to a privileged entry point within the driver thereby bypassing RTE. Thus, when a privileged driver is entered after an interrupt, RTE has *not* set up the tables or performed any of the processing that it performs before entering a standard driver. The driver is responsible for setting up the table addresses and any of the other processing normally done by RTE. When a standard driver returns to RTE, it is much like returning from a subroutine call. However, since RTE did not make the "call" to the privileged driver, the privileged driver cannot simply "return" to the OS. Before looking at the special processing required of a privileged driver, we will look at the situation that required us to use a privileged driver in the first place.

Why use a privileged driver?

RTE-A is shipped with interface drivers which control a wide variety of interface cards; none of them privileged. So it may be reasonable to wonder why a privileged driver is ever necessary. The answer is that for many realtime applications, it is critical to service an interrupt as quickly as possible. Except for the period where RTE is saving or restoring the state of the machine, privileged interrupts are always enabled. Non-privileged interrupts are only enabled when a user program is being executed. They are always disabled during RTE processing. This period can be on the order of milliseconds depending on the functions that the operating system is performing. For many realtime applications, this delay is excessive.

In our example of the golf swing, most professional golfers can swing a golf club in excess of 100 miles per hour. At this speed, the clubhead will travel 1 inch in 568 microseconds. For us to collect all of the data, it is critical that we detect the interrupt and read the data as fast as possible. If we don't, then we miss the swing. For many I/O transfers, the time constraints are much less stringent. For example, when reading a disk, the data will still be on the disk whether we get to the interrupt immediately or in the near future.

Golf Swing Analysis

Before going on to further discussion of privileged drivers, we will need to further clarify some of the needs of our specific application. When a golfer starts to swing the club, quite often the clubhead will move back and forth a bit before starting the actual swing. In golf parlance, this is called a waggle. It is important for the analyzer to distinguish between a waggle and a real swing. Thus, triggering is an important factor. Once the club has been taken all the way back and a downswing started, the plane along which the clubhead moves is critical. The clubhead can move along one of three paths: outside in, straight or inside out. At impact, the clubface can be open, straight or closed. The combination of these is what will determine the flight path of the ball. For example, an inside out swing with a closed clubface will cause the ball to draw (slight curve from right to left) or hook (severe curve from right to left). A slice (severe curve from left to right) is caused by bringing the clubfrom the outside in and leaving the clubface open.

After careful consideration (and some trial and error) we determined that building the data collector with the geometry used in Figure 2 suited our needs best.

The PIC card is a 16-bit parallel card. We were able to get 32-bits of sensitivity by tying the photocells in the left column to the corresponding photocell in the right column. During a normal golf swing, the 16-bits will start off at 0. As the club passes through the right column of photocells, some bits will get set. Because the distance between the left and right columns is greater than width of the head of the golf club, all bits will return to 0 as the club clears the right column. Then, more bits will get set as the clubhead crosses the left column. Finally, after the clubhead strikes the ball, all bits will return to 0 again.

The PIC card has 4 bits available for the swing analyzer to use as status bits. These are labeled ST0, ST1, ST2 and ST3. A feature of the PIC card is that ST0 can be programmed so that triggering it will generate an interrupt to the HP1000. We have tied two photocells to ST0 and two to ST1. When either ST0 cell is triggered, an unsolicited privileged interrupt is generated and our special interface driver ID55 is entered. ID55 then goes into a loop checking to see if an ST1 photocell has triggered. Our processing must take into account any waggle that may be underway. If the timing is not correct, the driver must then re-enable the ST0 trigger and exit the driver at the point at which the system was interrupted. If the timing is acceptable, the driver starts a DMA transfer of up to 32k readings of the photocells.

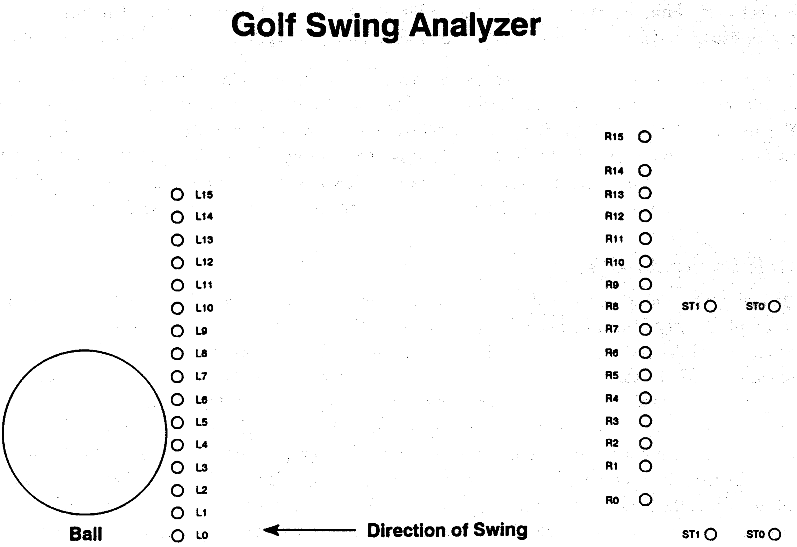


Figure 2

This is where some of the key distinguishing characteristics of our needs comes into play. The photocells put out a continuous voltage that to us looks like a 0 (not triggered) or a 1 (triggered). Our measurements require that we know the relative times that the various photocells are triggered. For example, if we know that R5 triggered at t1 and L5 triggered at t2, then the velocity of the clubhead can be calculated by taking the distance d(R5,L5) and dividing by the time difference (t2-t1). Likewise, the angle of the clubface at impact can be determined by taking the time differential between triggering cells L3 and L5. For example, if L5 triggers before L3, then the clubface is closed. The time difference tells us by how much.

Since we are taking data continuously from the start of the downswing until 32k samples have been collected, the question is how do we time tag the data? Normally, once a DMA transfer has begun, the driver will return control to RTE which will then determine the next task to perform. Under this scenario, when a DMA transfer is underway, the I/O processor (IOP) on the interface card is always competing with the CPU for memory cycles. When the IOP is granted control of the memory bus, it will transfer a data value directly from the interface card to memory. Then the IOP and CPU will contend for the memory bus again. This means that we cannot assume that the time period between any two data elements is a constant. In order to ensure a constant time value, our driver must begin a DMA transfer and then, instead of returning to RTE, it goes into a WFI (Wait For Interrupt) cycle (see Figure 3). This will hold off any other processing in the machine except for our data transfer.

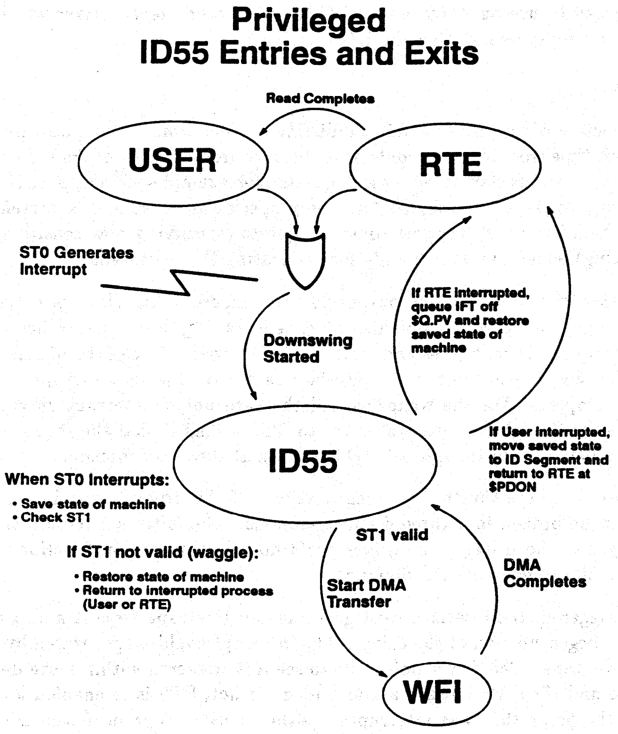

Privileged
ID55 Entries and Exits
Read Completes
USER
RTE
ST0 Generates Interrupt
Downswing Started
If RTE interrupted, queue IFT off $Q.PV and restore saved state of machine
If User interrupted, move saved state to ID Segment and return to RTE at $PDON
ID55
When ST0 Interrupts:
• Save state of machine
• Check ST1
If ST1 not valid (waggle):
• Restore state of machine
• Return to interrupted process (User or RTE)
ST1 valid
Start DMA Transfer
DMA Completes
WFI

Figure 3

Once CPU contention had been eliminated, there was still one other factor to think about. Because the A900 and A990 CPUs have a cached memory structure the time differential between any two data items collected on them may not be constant. For example, A990 cache lines are 32 bits and writing a 16 bit word to memory may result in a hit or a miss to 16 of the 32 bits. Depending on the circumstances, a write may result in no write to main memory, a write to memory or a read from memory then a write. The effect is that we can never be sure exactly what the time between any two data items is on a cached machine.

One other effect that we considered was the dynamic memory refresh cycles. Dynamic RAMs require being refreshed periodically. This varies from every 2 to 4 milliseconds depending on the size of the RAMs. The refresh only takes 1 memory cycle, however, so these effects can be ignored.

Thus, we determined that in order to obtain the most consistent time differential between samples, we needed to use an A600 or A400 CPU using a privileged driver which did not re-enter the operating system during the DMA transfer.

Privileged Drivers

The initiation section of our driver is fairly indistinguishable from a standard driver. Upon entry, the request type (read,write or control) is obtained from word 15 of the DVT ($DV15). Some drivers have various flavors of these requests (for example, binary, ASCII, Z-buffer etc.). Our driver, however, is designed for a very specific purpose and is therefore fairly simple. It can handle only a very few control requests (specifying how sensitive to make the trigger sensing) which are used mainly for calibrating the photocells.

The write function was added to send output to a bus mouse. This allows our PIC card to send the commands to a PC which look like someone is moving the mouse. When the driver receives a write request, it configures and starts the DMA write. One of the bits in the DMA control word specifies whether or not the interface card should generate an interrupt when the transfer is complete. For the write, we tell the card not to interrupt on completion. Thus, once we start the DMA write, we return to RTE telling it that the request has been successfully completed (even though it is taking place at that very moment).

The read request is where the true privileged aspect of the driver comes into play. The read request can be broken into three distinct sections. The initiation section is the non-privileged entry into the driver. The driver performs all initialization functions here and enables the ST0 bits to generate an interrupt.

When ST0 is triggered, the interface card generates an interrupt request and a branch is made to the privileged portion of the driver. The interrupt could be generated by a waggle or by a real downswing. The driver polls ST1 to see if it triggered within a pre-determined amount of time and if so, we initiate a DMA read. If not, ST0 is re-enabled and control is returned to the point that was interrupted (either a user program or the OS). To the interrupted process, the interrupt is totally transparent.

When the DMA read is started, all interrupts are disabled except those from the PIC card and the driver executes a .WFI instruction. The result is that the only activity on the system is the DMA transfer of the swing data. When the last word is transferred, the interface card generates an interrupt which again branches to the privileged portion of the driver. Because the driver was entered without knowledge of RTE, the driver must place the I/O transfer status information in a location known to RTE and then inform RTE that it has done so. At some future point, RTE will return control to the calling program and inform it that the requested transfer is now complete.

We will now look at the three sections of the read request in greater detail.

Read Initiation

The initiate section of the driver is entered from and returns to the operating system. The read request performs three functions: save any required parameters, set up the DMA transfer, enable the ST0 interrupt. Because the read initiation section is called by RTE all I/O table setup and mapping has been performed. During the initiation section, the driver performs all the set up and configuration tasks needed to execute the read, but it doesn't actually start the read. Instead, it enables the ST0 interrupt and returns to the operating system telling it that the read is underway. At that point, RTE determines which process (if any) to schedule next and normal system operations continue. When ST0 does trigger, indicating a potential downswing, the PIC card will generate an interrupt and the privileged section of the driver will be entered directly. That is, RTE will be bypassed and the tables needed for the I/O transfer will not be configured. Any pointers that need to be accessed during the privileged operation must be saved during the non-privileged initiate section. Our driver only needs to save the address of the 7th word of the IFT ($IFT7), the 16th word of the DVT ($DV16) and the starting address of the IFT extension ($IFTX) (a driver specific number of words beyond those allocated by default for an IFT). Local copies of these addresses are stored within the driver's address space.

The driver next configures the DMA transfer. That is, it sets the appropriate bits in the DMA control block which specifies how the DMA transfer should proceed. Normally at this point, the next step is to send the address of the DMA control block to the interface card and then issue the command to start the transfer. However, we are not yet ready to take the data. We configure it now because we won't have time to do it later when the actual downswing begins.

The DMA control block we use specifies an input transfer, in word mode, which will cause an interrupt when the requested number of words is transferred. Also, a Device Command Signal will be asserted for each data element transferred. The Device Command Signal is critical to our application. The interface card sets Device Command to signal the device that it is ready for the I/O transfer to begin. The device puts the data on the input lines and asserts the Device Flag Signal to indicate that the data is available. In our case, the data is always available and we are just waiting for the card to read it. Thus, we have tied Device Command to the Device Flag. When the interface card is ready to receive data, it sets Device Command. This in turn sets Device Flag which tells the interface card that the data is ready. The interface then latches the data on the input lines and saves it as an input value.

Before returning to RTE, the driver has one last function to perform. It must send a message to the PIC card telling it to generate an interrupt when the ST0 is set. We send a command to the PIC card that tells it to allow this bit to generate an interrupt when it is enabled.

```
LDA =b420
OTA 31b
STC 30b,c
```

As mentioned earlier, when the device is ready, it sets Device Flag which causes the card to generate an interrupt. The above instruction sequence causes Device Flag to get set when ST0 gets set. The STC instruction, however, stands for Set Control. This instruction is what enables the card to generate an interrupt. Because our device has the control and flag signals tied together, this has the effect of not only telling the card to allow ST0 to generate an interrupt, but it will generate an interrupt immediately unless we clear the flag signal. Before returning to the operating system we must issue the CLF 30b instruction which clears the flag and therefore the interrupt request.

Privileged Interrupt on ST0

When ST0 is triggered, a privileged interrupt is generated and the driver is entered directly. The driver first disables the interrupt system to prevent another event from causing us to lose this interrupt. Next, the driver checks to see if this interrupt was caused by ST0 being triggered or because the DMA read transfer completed. In order to determine this, the driver sets a flag when it starts the DMA transfer. When the DMA transfer begins, the ST0 interrupt is disabled. So, the next interrupt after the DMA transfer starts, must be the DMA completion. Before discussing the processing that must be done after the transfer completes, we will look at what happens during the ST0 interrupt processing.

When ST0 causes an interrupt, it could be a waggle or an actual downswing. The difference is determined by the time difference between setting ST0 and setting ST1. If this difference is not within the predetermined limits, then the process that was interrupted is resumed with the state of the machine set to exactly what it was when it was interrupted. This is done by saving the state of the machine when ST0 interrupts. The values that are saved are: the A, B, E, O, WMAP, C, Q, Z and global registers and the interrupt mask. Also saved are the point of suspension for the interrupted process and the RTE flag $MPTF. This flag will allow us to determine later whether it was RTE or a user process that was interrupted.

If it was a real downswing, then we are ready to start our DMA read. The driver issues a CLC 6 to turn off the TBG, starts the DMA transfer, turns the interrupt system on and executes the WFI instruction. The net effect is that our DMA transfer is the only thing happening on the system. When the DMA transfer completes, the interface card will generate an interrupt and the driver will be re-entered.

DMA Completion Interrupt

When the DMA completes, the driver must then tell RTE that the transfer completed and then transfer control to the operating system. There are two cases to consider: whether it was RTE or a user program that was interrupted by ST0. If it was RTE that was interrupted, then we must return control to RTE at the point at which it was executing prior to the interrupt. This is because RTE is not re-entrant. Before returning to RTE, though, the address of the IFT extension ($IFTX, which was saved when the read request was initiated) must be linked off the privileged driver done list headed by the system entry point $Q.PV. When we return, RTE will check this queue after it finishes processing its current task. To return to RTE, we follow the same procedure as returning from the ST0 interrupt above. The state of the machine that was save is restored and a branch to the interrupted location is executed.

If the interrupted process was a user routine, the return is much different. The state of the machine at interrupt reflects the state of the user program. The saved registers are stored in the program's ID segment, the interrupt system is enabled and a branch to the system entry point $PDON is made. When the branch to $PDON is made, the B-register is set to the IFT address for the interface card whose interrupt was just serviced. RTE then uses this address to determine which process initiated the I/O request on that interface and it later returns control to that process telling it that the transfer was successful.

Conclusion

The major differences between privileged drivers and standard drivers are in the services provided by RTE and in returning from the driver to RTE. Standard drivers provide most of the functionality of privileged drivers, but when the time between generating an interrupt and servicing the interrupt is critical, privileged drivers are the solution.

It should be noted that it may not be a good idea to disable all interrupts during a DMA transfer as we have. By using a logic analyzer, we were able to determine that we did achieve our desired results. The spacing between data items was a constant of ~1.6 microseconds. For a 32k transfer, this means that all interrupts, including the TBG, will be disabled for > 50 milliseconds. For many applications, this could be unacceptable. For other privileged drivers, it is often acceptable to keep privileged interrupts enabled, but increment the $MPTF flag. This allows the TBG tick to interrupt, but normal processing for the tick will be held off until after the privileged driver has completed its processing.

MANAGING MULTIPLE IDENTICAL RTE-A SYSTEMS:
A Customized Approach

Interex 1993
Paper# 1003

When supporting multiple RTE-A systems it becomes necessary to have a high degree of commonality among them, especially if geographically separated. To simplify this standardization, identification of locally customized system variables is necessary. Once this is done, a way to make these variables modifiable without making many repetitive changes to the system is necessary.

This paper describes a method where this commonality is obtained. It also briefly discusses a program developed to provide the link to these localized variables contained in a central configuration file.

Larry Ridgley
Hewlett Packard Company
1412 Fountaingrove Parkway, M/S 1BS-C
Santa Rosa, CA 95403
(707) 577-2155

9 July, 1993

INTRODUCTION

The project I manage (**MAXS+** by Combs International, Inc.) requires me to support over a dozen HP1000 systems at several out-of-area and out-of-state sites within the Hewlett-Packard organization. When a new release of the application software or the RTE-A operating system comes along, it requires efficient distribution to each user site. It must be standardized, needing minimum customization by the HP1000 system manager (some of whom have limited operating-system and/or programming knowledge). This paper describes one solution developed to handle this situation.

DEFINITION

Several tasks were identified that must be done to meet the goal of producing generic, customizable, bundled systems (these may be done in any sequence):

1. Define the variables. List all items that are uniquely different between systems.

2. *Define the constants.* This list should include those items that do NOT (or rarely) change. Group them by product or subsystem.

3. *Define standards.* Decide early-on what form the variables will take (name, style). Also define how they will be made available to both users and to the operating system. [That is what this paper is about.]

After making the first rough list of variables and grouping them by product or subsystem I came up with several unique items (this is only a partial list):

Product or subsystem: None; system identifiers
Variable and value: System ID = CPU0001
O/S rev. = 6000
Gen. rev. = CA

Product or subsystem: NS-ARPA/1000
Variable and value: Nodename = HPXYZZY
IP Address = 123.4.56.7
Domain = PLUGH.COM
Gateway IP = 123.4.0.1

A much longer list of constants was produced (just about everything else fit in this class!), so I'll list only a few examples: root directories and their locations (/SYSTEM, /PROGRAMS, /LIBRARIES,...), device LU numbers, etc. This list was informative, but useful only as a reference during the customization process.

With all of the unique variables identified, the next step involved standards definition. First, all of these variables are in a single flat text file that could be easily edited and viewed. Second, the syntax for all variable definitions is fixed as well. This resulted in the following criteria:

1. Filename and location: **/SYSTEM/ENVIRONMENT.VARS** (Type 4 file)

2. Comment lines start with '*' with blank lines allowed.

3. Variable names to be a minimum of 2 and a maximum of 16 characters long and could contain only alphanumeric and underscore characters; not case-sensitive. Also, they should *not* begin with a number (i.e., "**A1**" would be valid, but "**1A**" would not).

4. The value of a variable may be of any type (text or numeric), separated from the variable name by an '='. It may contain embedded blanks and punctuation and may be as long as the rest of the line allows (i.e., "**FOO = Hello, World!**").

5. The variables must be accessable by both CI and user programs.

The resultant file would be created by EDIT/1000 and made to look similar to the following figure:

```
*/SYSTEM/ENVIRONMENT.VARS <930601.0800>
* Global environment variables
*
* System information
ID                     = CPU0001
CPU_TYPE               = 990
RTE_REV                = 6000
GEN_REV                = CA
*
* NS-ARPA/1000 variables
NODENAME               = HPXYZZY
IPADDR                 = 123.4.56.7
DOMAIN                 = PLUGH.COM
GATEWAY_IP             = 123.4.0.1
```

Example **ENVIRONMENT.VARS** file

IMPLEMENTATION

After identifying the variables and placing them in the **ENVIRONMENT.VARS** file, it is necessary to write a program to access the file and the variables within.

The program is to provide the following (minimum) types of access to the variables:

1. Return the value of the variable specified in the runstring into the CI variable **$RETURN_S**.
2. Change the value of the variable specified in the runstring.
3. List the variables and their values to the terminal or printer.
4. Provide a user-callable routine to perform the same functions as items 1 and 2 above.

The resulting program is named **ENV.RUN**. This program is designed to be used in several ways:

RU,ENV,*opt* [,>*list*] [,<*fromfile*]

or

RU,ENV,*var* [,=,*value*] [,<*fromfile*]

where

opt =Option: ' ' or '?' gives help information.
'-l' shows list of variables in file.
'-q' quiet mode; no printed error msgs.
list =List device or file for '-l' option (leading '>' is required).
var =Variable name to get (or change) value of.
'=' =Required if variable's value is to be changed.
value =Value to change the variable to; may be text or numeric. If text string, be sure to put "s around it to prevent CI from modifying the string.
fromfile =If specified (leading '<' required) use this file for the variables instead of the default, **ENVIRONMENT.VARS**.

Returns:

$RETURN_S =Value of variable requested or changed.
$RETURN1 =Will be <0 if an error was detected; otherwise will be >=0 if successful.

ENV is intended for use within CI command files. The results of the call to get a variable's value is assignable to a CI variable (usually of the same name). Here is an example:

```
* Get system ID...
RU,ENV,ID
SET id = $RETURN_S
ECHO 'System ID is '$id
```

Example CI command file

Executing this command file will display: **System ID is CPU0001.**

Changing the value of an existing variable is also simple (NOTE: If a variable does not exist, an entry is appended to the file):

```
* Change system ID...
ASK 'New system ID?'
SET new_id = $RETURN_S
RU,ENV,-q,ID,=,$new_id
* Success?
IF IS $RETURN1 > = 0 -I
THEN
  ECHO 'System ID has been changed to '$new_id
ELSE
  ECHO 'System ID was NOT changed!  Error= '$RETURN1
FI
```

Example of changing a variable

Note that **ENV** returns a negative value in **$RETURN1** if an error occurs during variable access. An error message is displayed, unless the '**-q**' option is specified (as in this example).

CUSTOMIZATION

After all system variables have been defined and placed within the **ENVIRONMENT.VARS** file, one can now proceed to customizing the system to use them. To simplify maintenance of the system I have tried to limit this customization to command files. EDIT/1000 is used in batch mode to edit template files with the environment variable values.
Since the number of customized files may vary, depending on your needs, I have chosen one example that shows the flexibility of this method.

All that is required to customize a file is to have a current copy of the specific file with your values in place. Copy it into a template file. Now edit the template file, substituting a unique pattern for the actual value that you will be substituting for it later.

The following example initializes NS-ARPA/1000 (**WELCOME1.CMD** schedules **NS_INIT.CMD** at bootup). The original file was **/ETC/NSINIT.ANS**. The new template file is **/ETC/NSINIT.TPT**. The localized file is ***localsystemid*****_NSINIT.TEMP**.

Here is the process (only portions of the actual files are shown for brevity):

Original file:

```
* /ETC/NSINIT.ANS
:
2
HPXYZZY.PLUGH.COM
/D
:
N
123.4.56.7,255.255.248.0,LAN,,140,E
/D
:
```

Template file:

```
* /ETC/NSINIT.TPT
:
2
!nodename!.PLUGH.COM
/D
:
N
!ipaddr!,255.255.248.0,LAN,,140,E
/D
:
```

Here is a portion of the command file (**NS_INIT.CMD**) that copies the template file and then performs the batch edits of the resulting localized file:

```
* /CMDFILES/NS_INIT.CMD
* Initialize NS-ARPA/1000
:
WD /ETC
* Get System ID...
RU ENV id ; SET id = $RETURN_S
* Copy template file to localized file...
CO NSINIT.TPT $id\_NSINIT.TEMP D
* Get local NS-ARPA configuration info...
RU ENV nodename ; SET nodename = $RETURN_S
RU ENV ipaddr ; SET ipaddr = $RETURN_S
RU ENV gateway_ip ; SET gateway_ip = $RETURN_S
* Modify file with local values...
RU EDIT -q $id\_NSINIT.TEMP|1$x/!id!/$id//|er
RU EDIT -q $id\_NSINIT.TEMP|1$x/!nodename!/$nodename//|er
RU EDIT -q $id\_NSINIT.TEMP|1$x/!ipaddr!/$ipaddr//|er
RU EDIT -q $id\_NSINIT.TEMP|1$x/!gateway_ip!/$gateway_ip//|er
* Schedule NSINIT with localized file...
RU NSINIT $id\_NSINIT.TEMP $id\_NSINIT.ANS NSINIT.LST
:
```

Example using variables for customizing

The actual file created from the template (in our example system) would be **/ETC/CPU0001_NSINIT.TEMP**. Continue in this same manner with any of your files that need localization.

SUMMARY

This customized, centralized approach to managing multiple systems is of great benefit. Much time (time = $$$) and effort is saved by this method. The **ENV** program is a powerful tool that allows centralized control of global environment variables. (NOTE: The **ENVIRONMENT** program and associated modules are available through the **INTEREX** CSL/1000 distribution channels.) Used with the enhancements to RTE at 6.0 (an EVB for each user) makes a very versatile combination. The use of these ideas is limited only by your imagination.

* * *

PAPER NUMBER: 1005

TITLE: Using Modems on the HP 1000 A Series Computers

PRESENTER:

Alan Tibbetts
Consultant
3498 Gibson Ave.
Santa Clara, CA 95051
408-247-7280

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Standards-based Networking Services on the HP 1000

Lynn Rodoni
Mydung Tran

Hewlett-Packard Company
Software Technology Division (SWT)
11000 Wolfe Road
Cupertino, CA 95014

This paper discusses the industry standard networking services available on the HP 1000. General descriptions of FTP and TELNET as well as detailed descriptions of enhancements in the 6.0 release of RTE-A will be provided. These will cover the areas of functionality, interoperability, RTE unique features, and performance. The addition of Inetd into the networking services model will also be presented.

1. INTRODUCTION

Resource sharing and information exchange are significant features of computer networking. NS-ARPA/1000 is a data communication product that enables HP 1000 systems to share access to resources such as disc files, printers, magnetic tapes, terminals, etc. with other computers. The layered design architecture of NS-ARPA/1000 offers a structured, modular approach to the different tasks that have to be performed in order to transmit and interpret data across a network.

At the top of the model are *User Services* such as file transfer, remote command execution, and remote file access. There are no separate Session Layer, Presentation Layer, or Application Layer within the NS-ARPA/1000 and ARPA/1000 architectures. Rather, each service incorporates the appropriate networking protocol within its individual implementation. This has historically been a common approach to network implementations.

Below is a summary of the different services available on the HP 1000. The ARPA Services are available in both the ARPA/1000 and the NS-ARPA/1000 products. The remaining services are only available in the NS-ARPA/1000 product. DS Networking and NS Networking were developed as proprietary solutions and made available on HP systems including the HP 1000. Proprietary solutions are too restrictive for many within today's world. TCP/IP and ARPA Services have become the ***defacto standard*** within the industry and are supported by the HP 1000 in the NS-ARPA/1000 and ARPA/1000 products. In addition, there are a number of services developed at UC Berkeley called the Berkeley Services which are often found along with the services defined by ARPA. At this time, Berkeley Sockets or BSD IPC is available with the NS-ARPA/1000 product.

a. ARPA Services: TELNET, FTP
b. Berkeley Services: BSD IPC
b. NS Common Services : NFT, NetIPC, RPM
c. DS/1000-IV Compatible Services: these services are also part of the DS/1000-IV product (the predecessor to NS-ARPA/1000)
  - RTE-RTE Services that can used for backward compatibility with DS/1000-IV as well as for NS-ARPA/1000 to NS-ARPA/1000 communication.
  - Transparent File Access (TRFAS), part of RTE-RTE services, also known as DS File Transparency, which allows users to access HP 1000 remote files using RTE file manipulation commands.
  - RTE-MPE services that can be used for backward compatibility with DS/3000 as well as for NS-ARPA/1000 to NS3000/V communication.

The *Transport Layer* handles end-to-end communication between source and destination systems. TCP is the ARPA transport protocol and NetIPC or Berkeley Sockets provide programmatic access to the network at the Transport level.

The *Network Layer* (also referred to as the IP layer) is responsible for addressing functions. It makes sure that packets of data are acquired by the system to which they are addressed.

The actual transmission of the data over the communication link is governed by the *Data Link Layer*. This layer is a combination of IO card and driver on the HP 1000. They work together to send and receive data in useful chunks called packets.

The lowest layer, the *Physical Layer*, provides electrical and mechanical specification for the transmission of bits across the link. This corresponds to the LAN cable or other means of physically transmitting the data from one system to another.

Figure 1 illustrates the overall NS-ARPA/1000 architecture.

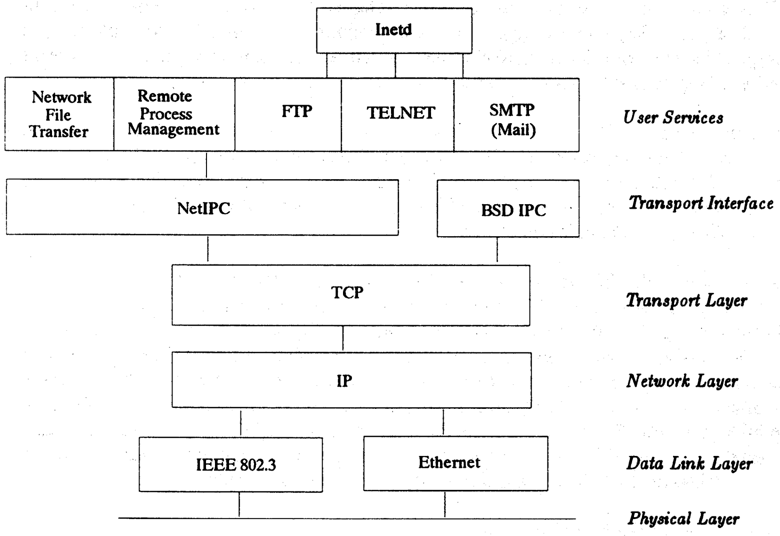

Inetd
Network File Transfer
Remote Process Management
FTP
TELNET
SMTP (Mail)
User Services
NetIPC
BSD IPC
Transport Interface
TCP
Transport Layer
IP
Network Layer
IEEE 802.3
Ethernet
Data Link Layer
Physical Layer

Figure 1: NS-ARPA/1000 Architecture

NFT (Network File Transfer) and RPM (Remote Process management) are ***proprietary services*** which are available over TCP/IP. These services are supported for compatibility with other HP systems. SMTP is a standard protocol used by electronic mail facilities in a TCP/IP network. It is used by MAIL/1000. Berkeley Sockets (BSD IPC) is an interface to the Transport Layer and is discussed in detail in the **NS-ARPA/1000 BSD IPC Reference Manual** as well as **BSD IPC on the HP1000,** a paper published as part of the San Diego INTEREX Proceedings, 1991. NetIPC is an HP proprietary interface to the Transport Layer. These services and interfaces will not be discussed within the rest of this paper.

The main focus of this paper is on the User Services, specifically those which are a part of the ARPA Services. These are **FTP** and **TELNET** which are both included in ARPA/1000 and NS-ARPA/1000. **Inetd**, a general monitoring facility for user services, will also be discussed. This monitor has been incorporated into NS-ARPA/1000 and ARPA/1000 as of the 6.0 Release and replaces previous individual monitors for FTP and TELNET.

2. FTP

Description

FTP stands for File Transfer Protocol and allows authorized users to log into a remote system, identify themselves, perform file management operations, such as changing, listing, creating and deleting remote directories. Simple text or executable binaries can be transferred via FTP reliably and efficiently. FTP shields users from variations in file storage systems among hosts.

FTP is used to transfer files interactively or programmatically. Users can extract and deposit files from one system to another via the network rather than using tapes or other physical media to transport data from one geographical location to another. The saving in time is great especially when the two locations are quite far apart. FTP simplifies the file transfer task a great deal. With FTP, one computer can act as the go-between for two other computers. In other words, users can initiate a transfer between two computers other than the host. A special service, "Anonymous FTP", is easily used to distribute software. Users can deposit whatever files they want to make available to others into a special directory.

Example

The following example describes a session of FTP in which multiple files are transferred from an *HP 1000* computer to an *HP 9000* system. At the system prompt (system1>) the user invokes the FTP program specifying the source system. The source or remote system is where the files to be transferred reside. The user is prompted to enter the user account and password on the source system. If the correct information is entered, the login process completes successfully and the user can change the destination directory (where the files will be deposited) and the source directory (where the files to be copied reside). In this example a source directory listing is requested to confirm the existence of the desired files (nssys libraries, cds and non_cds versions). The user sets the transfer mode to binary and initiaties the transfer. Since this is a multiple file transfer, FTP validates each file before transferring it. Finally, the user exits the FTP program.

Below is the dialog representing the example described above. The text which the user specifies is in italics and underlined. Comments describing the specific actions/results are indented and are not a part of the actual dialog.

```
system1> ftp system2
Connected to system2.
220 FTP/1000 Rev. 6000 Service ready for new user.
```

System1 and system2 are now connected. Access to system2 must be validated.

```
Name ( system2: mydung)
331 User name okay, need password.
Password:
230 User logged in, proceed.
Remote system type is RTE-A.
```

The user has successfully logged in to system2 which is recognized as an RTE-A system.

```
ftp> lcd/tmp
Local directory now /tmp
```

The local directory (on system1) is now /tmp.

```
ftp> cd/libraries
250 CD  command successful.
```

The remote directory (on system 2) is now /libraries.

```
ftp> ls ns@
200 PORT command successful.
150 Opening data connection for file list.
total 4906
-rw-r--r-- 1 system  system   601344    Dec 9 1991 nssys.lib
-rw-r--r-- 1 system  system   654080 Dec 9 1991 nssys_cds.lib
-rw-r--r-- 1 system  system   654080 Dec 9 1991 nssys_cds_s.lib
-rw-r--r-- 1 system  system   601344 Dec 9 1991 nssys_s.lib
226 Closing data connection.
```

A listing of the interesting source files (on system 2) is requested and displayed.

```
ftp> bin
200 Type set to I.
```

The transfer mode is set to binary.

```
ftp> mget nssys@
```

The user requests a transfer of multiple files be initiated.

```
mget nssys.lib?y
```

FTP asks the user to validate the transfer of the first file it finds matching *nssys@* in the source directory.

```
mget nssys.lib?y
```

FTP asks the user to validate the transfer of the second file it finds matching *nssys@* in the source directory.

```
200 PORT command successful.
150-Opening BINARY mode data connection for
150  NSSYS.LIB:::5:2350:128
226 Closing data connection.
601346 bytes received in 13.48 seconds (43.58 Kbytes/s)
```

FTP completes the transfer and reports statistics regarding the transfer to the user directory.

This dialog continues until all files selected are transferred.

```
mget nssys_cds.lib?y
200 PORT command successful.
150-Opening BINARY mode data connection for
150  NSSYS_CDS.LIB:::5:2556:128
654082 bytes received in 17.93 seconds (35.63 Kbytes/s)
```

```
mget nssys_cds_s.lib?n
```

Here is an example of a file the user does not want transferred.

```
mget nssys_s.lib?n
ftp>bye
221 Service closing control connection.
```

The user exits FTP.

The FTP model

For NS-ARPA/1000, FTP allows users to transfer files among HP 1000, HP 3000, HP 9000, HP Vectra PC, IBM PC, Sun, VAX, and other computers which support the transport and routing protocol TCP/IP. The ARPA Services on the HP 1000 use standards defined by the Advanced Research Project Agency (ARPA) and FTP uses ARPA Standard File Transfer Protocol.

The FTP model, illustrated in Figure 2, consists of two programs, the client and the server. There are two processes associated with each program, the control connection process and the data transfer process. Two separate TCP connections are established. The control connection connects the client control process with the server control process which carries commands. The data transfer connection connects the client data transfer process with the server data transfer process which carries all data.

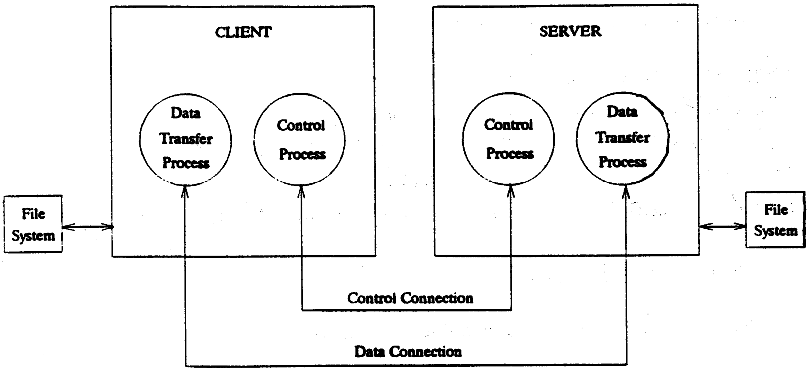

CLIENT
SERVER
Data Transfer Process
Control Process
Control Process
Data Transfer Process
File System
File System
Control Connection
Data Connection

Figure 2: The FTP Model

List of commands

The commands currently supported as of the 6.0 Release are listed in the table below. Note that all commands, other than those with an asterisk (*) in the **RTE Only** column, are supported by HP-UX FTP Services with the identical syntax. This provides for interoperability at the user level with a minimum amount of 'special' commands to remember.

Command	RTE Only	Description
!	*	Invokes CI on the local host.
? or ??		Displays FTP commands and help information. Same as HELP.
..	*	Sets the working directory on the remote host to the parent directory, i.e. one level above the current one.
/	*	Displays the FTP command stack.
APPEND		Transfers local file to the end of remote file
ASCII		Sets the FTP file transfer type to ASCII
BELL		Sounds a bell after each file transfer completes.
BINARY		Sets the FTP file transfer type to BINARY.
BYE		Closes the remote connection and exits from FTP. Same as EXIT and QUIT.
CD		Set the working directory on the remote host to the specified remote directory.
CLOSE		Closes the remote connection and remains in FTP.
DEBUG	*	Prints commands that are sent to the remote host.
DELETE		Deletes the specified remote file or empties the remote directory.
DIR		Writes an extended directory listing of a remote directory or remote file to the terminal or to a local file.
DL	*	Writes an extended directory listing in RTE-A DL format to the terminal or to a local file.
EXIT	*	Closes the remote connection and exits from FTP. Same as BYE and QUIT.
FORM		Sets the FTP file transfer form to the specified format. The only supported format is non-print.
GET		Transfers remote file to local file. Same as RECV.
GLOB		Toggles file name globbing (expansion) for multiple file operations.
HASH		Toggles hash-sign (#) printing for each data block transferred.
HELP		Displays FTP commands and help information. Same as ? and ??.
LCD		Sets or displays the local working directory.
LL	*	Specifies a log file to which FTP sends the commands and miscellaneous messages ordinarily displayed to the user's terminal.
LS		Writes an extended directory listing of a remote directory or remote file to the terminal or to a local file.
MDELETE		Deletes multiple remote files.
MDIR		Writes an extended directory listing of remote directories or remote files to a local file.

MGET		Transfers multiple remote files to the local system, using the same file names.
MKDIR		Creates a remote directory.
MLS		Writes an abbreviated directory listing of remote directories or remote files to a local file.
MODE		Sets the FTP file transfer mode to the specified mode. The only supported mode is stream.
MPUT		Transfers multiple local files to the remote system, using the same file names.
NLIST		Writes an abbreviated directory listing of a remote directory or remote file to the terminal or to a local file
OPEN		Establishes a connection to the remote host.
PROMPT		Toggles interactive prompting.
PUT		Transfers local files to remote files. Same as SEND.
PWD		Writes the name of the remote working directory to the terminal.
QUIT		Closes the remote connection and exits FTP. Same as BYE and EXIT.
QUOTE		Sends arbitrary FTP server commands to the remote host.
RECV		Transfers remote file to local file. Same as GET.
REMOTEHELP		Requests help information from the remote host.
RENAME		Renames a remote file or remote directory.
RMDIR		Deletes an empty remote directory.
RTEBIN	*	Sets the FTP file transfer type to BINARY. PUT will create destination file names with the full RTE file descriptor.
SEND		Transfers local file to remote file. Same as PUT.
SITE		Performs server specific services.
STATUS		Writes the current status of FTP to the terminal.
STRUCT		Sets the FTP file transfer structure to the specified structure. The only supported structure is file.
SYSTEM	*	Shows the remote system type.
TR	*	Specifies an input file from which to get FTP commands.
TYPE		Sets the FTP file transfer type to the specified type. ASCII and BINARY are the types currently supported.
USER		Logs into the remote host on the current connection, which must already be opened
VERBOSE		Toggles verbose output. When verbose output is enabled, FTP displays responses from the remote host.

3. TELNET

Description

The TELNET protocol is a standard ARPA service that provides a virtual connection to a remote system on the network. It enables a user to logon to a remote system as if he/she was on a terminal directly attached to the remote system. The user enters commands and receives responses at the local terminal just as if the user's session were local. Input and output to the local terminal pass through a "virtual" terminal configured on the remote system. The remote commands are transmitted over network connections, sent to the virtual terminal, and subsequently executed on the remote system.

To connect to a remote host that is known on the user's network, the user simply invokes TELNET specifying the name of the desired remote system as an argument. Users can also chain several TELNET sessions. Chaining makes it possible to hop across the network to different hosts, which is not known directly within the user's machine subnet but via gateways, i.e. systems that are attached to two or more networks.

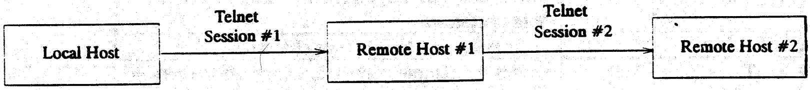

Local Host
Telnet
Session #1
Remote Host #1
Telnet
Session #2
Remote Host #2

Figure 3: Chained Telnet sessions to reach a distant host

The TELNET model

Similar to FTP, TELNET also consists of two programs, the client on the user's machine and the server on the remote system. A TCP connection is established between the client and the server. Keystrokes are typed on the user's terminal, accepted by the client program, and sent over the connection to the server. The server sends back characters and the client displays them on the user's terminal.

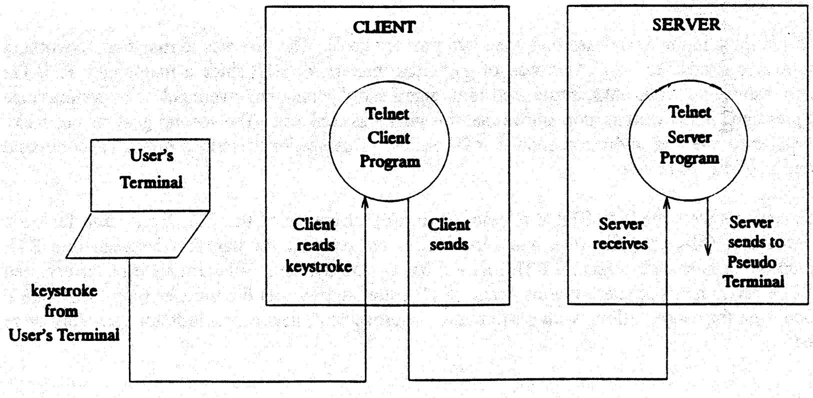

CLIENT
SERVER
Telnet
Client
Program
Telnet
Server
Program
User's
Terminal
Client
reads
keystroke
Client
sends
Server
receives
Server
sends to
Pseudo
Terminal
keystroke
from
User's Terminal

Figure 4: The Telnet Model

TELNET commands

TELNET has 12 commands which are listed below.

Command	RTE ONLY	Description
?		Displays TELNET commands and help information . Same as HELP
CLOSE		Closes the remote connection and logs off the remote session.
ESCAPE	*	Defines the TELNET escape character.
EXIT	*	Closes the remote connection, logs off the remote session, and terminates TELNET. Same as QUIT.
HELP	*	Displays TELNET commands and help information. Same as ?.
INTERRUPT	*	Changes the TELNET remote interrupt character.
MODE		Changes the data transmission to either line or character mode.
OPEN		Establishes a connection to a remote host.
QUIT		loses the remote connection, logs off the remote session, and terminates TELNET. Same as EXIT.
RUN	*	Runs a local program.
SEND		Sends special characters or commands to the remote system..
STATUS		Displays status of the TELNET remote connection.

4. New 6.0 features

For networking in the 6.0 release ther were two primary goals. The first was to maintain networking performance relative to 5.24. This was of particular interest for FTP since a major new RTE file system feature, symbolic links, introduced some significant file handling overhead. The performance data presented in the next section shows that this goal was achieved. The second goal was to make FTP easier to use and address particular RTE issues. This section describes new FTP command sadded as of the 6.0 release.

Enhancements were made to FTP at release 6.0 to take advantage of the RTE file system features. File attributes (file type, file size, and record size) are retained for transfers between one RTE revision 6.0 or later and a second RTE revision 6.0 or later system. Whenever a transfer between two RTE revision 6.0 or later systems occurs, FTP automatically sets the transfer type to BINARY for better performance. Along with performance improvement, new commands for useability were added :

DL this command requests an RTE-A format directory listing from a revision 6.0 or later FTP HP-1000 server, therefore, it only works when a revision 6.0 or later FTP client communicates with a revision 6.0 or later FTP server.

NLIST provides an abbreviated directory listing. The following table shows the FTP commands available for listing remote directories and files:

Listing a Single Directory/File		**Listing Multiple Directories/Files**	
DIR	extended listing	MDIR	extended listing
DL	extended RTE listing		
LS	extended listing	MLS	extended listing
NLIST	abbreviated listing		

Examples

1. To obtain information such as protection mode, owner, file size, time stamp of files, invoke the *dir* command in FTP.

```
ftp> DIR
200 Type set to A.
200 PORT command successful.
150 Opening data connection for file list.
total 3
-rw-r–r–  1 manager  system     628 Sep 30  1992 inetd.conf
-rw-r–r–  1 manager  system     532 Sep 30  1992 services
226 Closing data connection.
147 bytes transferred in 0.21 seconds  [ 0.70 kbytes/second ]
200 Type set to I.
```

2. To get RTE file attributes, the *dl* command is the correct one for FTP.

```
ftp> DL

200 Type set to A.
200 PORT command successful.
150 Opening data connection for file list.
directory /INT1/ETC
name            ex    prot  type blks words recs  addr/lu
NETD.CONF       rw/r  /r    4    3     314   13   327760/16
SERVICES        rw/r  /r    4    3     266   12   328320/16
226 Closing data connection.
204 bytes transferred in 0.21 seconds  [ 0.97 kbytes/second ]
200 Type set to I.
```

3. For UNIX users who happen to work on RTE, the *ls* command is more familiar than the *dir* command.

```
ftp> LS
200 Type set to A.
200 PORT command successful.
150 Opening data connection for file list.
total 3
-rw-r–r–  1 manager  system      628 Sep 30  1992 inetd.conf
-rw-r–r–  1 manager  system      532 Sep 30  1992 services
226 Closing data connection.
147 bytes transferred in 0.23 seconds  [ 0.63 kbytes/second ]
200 Type set to I.
```

4. If one is only interested in the list of the files and does not care about other file attributes such as size, time stamps..., the *nlist* command provides a faster response especially when the directory contains a large number of files.

```
ftp> NLIST
200 Type set to A.
200 PORT command successful.
150 Opening data connection for file list.
inetd.conf
services
226 Closing data connection.
22 bytes transferred in 0.17 seconds  [ 0.12 kbytes/second ]
200 Type set to I.
```

RTEBIN	This new command is specific to RTE. It has two functions. It sets the transfer type to BINARY. It also causes FTP to add the file type, size and record length to the destination file descriptor(s) when the user does a PUT or MPUT, thus retaining this information in their file names on non-RTE-A systems. This command is recommended when using PUT or MPUT from a revision 6.0 or later to a pre-revision 6.0 RTE HP 1000 system to preserve the file attributes and improve performance.
SITE	This command is used to pass commands that request server-specific functions. The user must use a REMOTEHELP SITE command to list the functions that the server supports.
SYSTEM	The server will respond with its system type when this command is used. When FTP knows that the server is an HP 1000, it will set the transfer mode to BINARY and transfer the file type, size and record length along with the file.

Inetd

Inetd has been added to NS-ARPA/1000 and ARPA/1000 to replace FTPMN (FTP Monitor) and TNMON (TELNET Monitor). Inetd is the monitor that listens for incoming FTP, TELNET, and Mail/1000 connection requests and schedules the appropriate server to handle the connection. Similar to the inetd super daemon in UNIX, inetd must be running before other hosts can connect to the local host through mail, ftp, or TELNET. Inetd is scheduled by NSINIT. Inetd also offers an extra level of security by allowing users to specify which hosts may or may not use a service. With inetd as the only monitor that can listen to many servers, system resources such as number of processes and the system load are reduced. Pre-6.0 revision, three processes (ftpmn, tnmon, and inetd) were required to monitor incoming request for ftp, telnet, mail services. With 6.0 or later revision, only one process, inetd is required.

5. 6.0 Performance Data

The following table summarizes FTP performance results on three different platforms, the A400, the A900, and the A990.

FTP Throughput
KByte/sec

CPU	Release 5.2	Release 5.24	Release 6.0	Improvement 5.2 to 5.24	Improvement 5.24 to 6.0
A400	7-15	10-21	10-21	40-43%	(3)-0%
A900	16-21	23-43	---	39-44%	---
A990			44-75	N/A	N/A

FTP on the A400 experienced a slight degradation (3%) only during PUTs using the ASCII transfer mode. BINARY mode is the preferred transfer method in any event since it always results in a faster transfer rate.
Networking on the A400 is the lower limit in all cases. Of more interest to many is the A990 which delivers twice the throughput of the A900 on 5.24. In some cases it is even better. 6.0 FTP on an A990 is almost a factor of 4 faster than 5.2 FTP on an A900. This is a combination of improved CPU performance and improved networking software.

The following table summarizes TELNET performance results on three different platforms and three software releases.

TELNET Transfer Rate
Char/sec

CPU	Release 5.2	Release 5.24	Release 6.0	Improvement 5.2 to 5.24	Improvement 5.24 to 6.0
A400	2247-2659	3104-3494	3350-3750	31-38%	7-8%
A900	4190-4588	5330-5797	---	26-27%	---
A990			8650-9300	N/A	N/A

As can be seen, the A400 TELNET performance continued to improve by 7-8% in the 6.0 release. Another way to look at TELNET is to compare the performance of a TELNET connection to that of a terminal connected directly to the system. The Transfer Rate in this case is roughly 5800 char/sec for all platforms. This means that a TELNET connection to an A990 will see better performance than a direct connect terminal!

6. Summary

Without data communication products, a user is limited to using one computer system at a time unless he/she is surrounded by several terminals each connected to a different system. Networking products provide the user with the capability of sitting in front of only one screen able to connect to any computer known to the user's network. Resources are more efficiently utilized via the network, i.e. many computers can share disks, printers, tape devices ....

ARPA Services and TCP/IP have become a defacto standard across many platforms in the industry. The HP 1000 incorporates these in two products, ARPA/1000 and NS-ARPA/1000. The primary differences between these two products is that NS-ARPA/1000 includes a number of services and transports for compatibility with other HP proprietary networking and programmatic interfaces to the transport.

With the 6.0 release, FTP, the ARPA service for transferring files, was enhanced to support new and existing features of the RTE file system. This was done in three ways. The first was to automatically select the optimum transfer mode for RTE to RTE transfers between systems running 6.0 and later revisions. The second was to add the command RTEBIN to preserve RTE file attributes when transferring files using a UNIX system as an intermediary. Finally, FTP supports the new file system feature, symbolic links.

BIBLIOGRAPHY

1. NS-ARPA/1000 User/Programmer Reference Manual
2. NS-ARPA/1000 BSD IPC Reference Manual
3. Postel, J. and J.Reynolds, RFC 959 File Transfer Protocol, ISI, October 1985.
4. Comer, D. E., Internetworking with TCP/IP Volume I, Prentice-Hall, Inc. 1991.
5. Rhadakrishnan, R., BSD IPC on the HP1000, San Diego INTEREX Proceedings, 1991.

An HP-UX Compatible Spooler for RTE

Todd Poynor

Hewlett-Packard Company
Software Technology Division (SWT)
11000 Wolfe Road
Cupertino, CA 95014

The forthcoming 6.1 release of RTE-A/VC+ provides a new printer spooling system that offers interoperability with the HP-UX spooler. This paper aims to introduce the spooler at a very high level, as well as discuss a number of design choices made in the HP 1000 implementation.

Background on the New Spooler Project

This paper will plunge into a description of the new spool system all in good time. But first, we try to begin at the beginning by presenting some of the issues that influenced the overall design of the system.

The existing RTE-A/VC+ spooler, that is, the **SP** program and friends, is among today's most glaring examples of RTE functionality in a state of disrepair. A perennial least-favorite among customers and HP personnel, the strikes against it include both a sizable collection of outstanding defect reports and a lack of flexibility to accomodate the many enhancements requested of it. In addition, long-time RTE users are fond of pointing out that the RTE-A spooler lacks significant functionality included in its predecessors on RTE-IVB and RTE-6.

HP has indicated for some time now that we intend to take action on the spooling situation. At the 1992 INTEREX conference we solicited suggestions on spooler improvements and future directions from the INTEREX membership [1], with the implication that your feedback would be addressed in a future RTE release.

Rationale For A New Spool System

It is possible that the existing spool system could have been brought up to a reasonable level of quality through major revamping, but concern for backward compatibility suggested that we leave that system more or less intact. It is difficult to envision the many needed modifications being made without adversely affecting established applications in some manner. For instance, the layout of the basic data structure used to keep track of spooling operations is documented. This data structure may be passed to customer-written software by a supported means, thereby forbidding any major changes to it if some semblance of backward compatibility is to be maintained.

---

[1] *Open Discussion on Spooling and Graphics Enhancements*, moderated by Scott Anderson and this author. Transcript available from INTEREX.

Additionally, we wished to implement a "standard" user interface and networked spooling strategy, as detailed in a following section. The existing spooler suffers the twin drawbacks of a clumsy interactive interface and a programmatic EXEC call interface (via the **SMP** program) that is quite limiting.

For these reasons, we chose to develop a new system that can execute concurrently with the existing system. This is not to imply that the old spooler is neglected in the 6.1 release; indeed, a large percentage of the outstanding defect reports and enhancement requests are addressed in the release. Furthermore, the old spooler still occupies an important position in the RTE-A/VC+ product, since it provides functionality not covered by the new system, as we shall see.

Design Guidelines for the New Spooler

From the outset, it was decided that the new spool system would be geared toward the spooling of output destined for printers. This is by far the most common usage of the existing system, and the subject of most enhancement requests we receive. Nonetheless, any device handled by the existing system may be serviced by the new system through custom device handlers.

Certain "exotic" aspects of the old system are not addressed in the new spooler. These include:

- The "redirection of I/O between LUs" feature.
- The "redirection of I/O bound for an LU to a file" feature.
- The "system error logging" feature.

All of these features are felt to be reasonably well covered by the existing implementation, as enhanced in the 6.1 release. The argument may also be made that these areas are not strictly within the domain of a spooling system, when defined as a mechanism for coordinating access to shared peripherals among multiple users.

The concept of "inspooling" is not addressed. To some people, this term suggests redirection of input between LUs and files. This notion is usually then expanded into UNIX[2]-style shell command line I/O redirection, which is, in turn, tied to the definition of "standard input and output" files for processes. The solution to these problems quickly balloons into a much larger issue than the spool system complaints we originally proposed to tackle. The term "inspooling" is also sometimes used to mean input batch job processing as provided by the RTE-IVB and RTE-6 Batch and Spooling System. Batch processing remains of importance to a few customers, but is a much less serious concern for the vast majority of RTE users than is printer outspooling.

---

[2] UNIX is a trademark of UNIX Systems Laboratories, Inc. in the U.S. and other countries.

Coexistence of the Old and New Systems

Because the old spooler provides needed functionality not offered by the new system, and because of the usual goals of backward compatibility, both the new and old spool systems must be able to execute concurrently. Both systems must thus be able to attempt to send output to the same devices concurrently without incurring "interleaved output", using LU locking to gain exclusive control of the device during output.

A Paradigm for the New Spooler

Given the decision to provide a new spool system alongside the old, we then had to decide on an overall system design: how should the spooler operate, and what user interface should be provided? Should we invent Yet Another Spool System from scratch, fine-tuned for the RTE environment, or should we base our new work on existing models?

HP, in recent years, has tended to pattern new RTE functionality after existing UNIX functionality. For this project, we were particularly pushed in that direction by the requirement that the new design encompass remote spooling between RTE and HP-UX. Hence, it should come as no great surprise that we required the user interface to be modelled after a UNIX interface. Some seasoned RTE veterans might be happier had we copied the RTE-IVB spooler design instead. But the HP 1000 is no longer an island; compatibility and interoperability with current networked services on other machines is of paramount importance these days.

Unfortunately, there is considerable debate over the relative merits of the various spool systems commonly used on UNIX. There are two longtime mainstays:

- The Berkeley Software Distribution (BSD) spooler (**lpr**, **lpd**, etc.).
- The AT&T System V spooler (**lp**, **lpsched**, etc.). A modified version of this spooler is currently shipped with HP-UX.

Both systems are in fairly wide disrepute for various reasons. Both receive criticism for a lack of robustness (a familiar theme to the RTE-A community when it comes to spoolers). Both are widely considered difficult to use, although "friendly" interfaces, such as **SAM** on HP-UX, alleviate this problem somewhat. The BSD spooler takes additional hits for a lack of flexibility and for only supporting relatively simple operations to be performed. One of the most frequently-heard complaints is that this system has precious little support for passing device-specific options to spooled device handlers. Device-dependent options are typically specified by embedding the proper escape or control sequences into the spooled file, making forms control and font management something of a headache. Debate continues as to whether this sort of formatting is the proper domain of the spooler or whether it should be left up to the utilities that produce the spooled output; no undisputed best answer has been found.

One saving grace of the BSD spooler is that it supports network printing, whereas the stock System V spooler does not. HP-UX and other System V implementations transcend that limitation by grafting BSD-style network printing onto the System V spooler. The union of the two is less than seamless, but is serviceable.

These two spoolers are not the only systems in use on UNIX. Other spoolers have been developed and continue to be developed that attempt to correct the deficiencies of the Big Two spoolers. When deciding the strategy for the new RTE spooler we chose not to try to predict the future directions in spooling that are likely to become prevalent in the computer industry. Our customers need solutions now that are of use in present-day networks and that are based on established models. Thus, we restricted our choice of models to the Big Two.

While neither of the Big Two UNIX interfaces presented a clear winner, we felt that our best strategy was to provide user interface compatibility with HP-UX. Therefore, we chose to implement an interface very similar to the System V spooler, also implementing the BSD networking extensions that HP-UX provides. This decision today seems fortuitous: it now appears that the industry trend is moving away from BSD in favor of System V. Longtime BSD stalwart Sun Microsystems has recently converted to a System V operating system (and spooler) with a BSD compatibility layer on top.

Introducing the LP Spool System

The new spooler is named the "LP spool system", somewhat in keeping with System V terminology. Confusingly enough, there are now three spoolers that run on RTE-A:

- The "SP spool system". This is the "old spooler", historically known as the "VC+ spooler", that we discussed in the previous section. Most of the printer spooling capabilities of this system are superceded by the LP spool system, but are retained for backward compatibility. The SP spool system continues to provide the only support for LU redirection, diverting output from an LU to a file, system error logging, and built-in support for magnetic tape devices.
- The RTE-A **PRINT** and **PRINO** programs, which are dedicated to spooling files to printers. These programs are not usually thought of as a spool system, but a spooler is, in effect, what is implemented. This is the only spooler available on RTE-A without the VC+ system enhancement package.
- The new LP spool system, which spools files to local printers and plotters, and which supports network printing between UNIX hosts and other RTE systems via NS-ARPA/1000 or ARPA/1000.

Each of the above spoolers may execute concurrently without interfering with the operation of the other systems. Because the output device LU is locked to the program performing the output, all three systems may simultaneously attempt to access the same device without disrupting the output of another system. It was not necessary to modify the existing spoolers to coordinate their device access with the new system. The LU locking feature of RTE turns out to be quite useful in this regard.

As discussed previously, the user interface to the LP spool system is very similar to that of the HP-UX LP spooler. People who are already familiar with the HP-UX spooler or with another implementation of the System V spooler will be instant experts on the RTE implementation.

This new spool system is intended to be a replacement for the printer spooling capabilities of the existing spoolers, both the SP and PRINT systems. Future HP efforts at improving printer spooling functionality are expected to be focused on the LP spooler.

The rest of this section introduces LP spool system concepts at a rather high level.

LP Spool System Overview

The LP spool system consists of several programs, the majority of which present a user interface that is very similar to the LP spool system that runs on HP-UX systems. All programs in the system take commands only from the runstring; there are no interactive programs. Therefore, the "programmatic" and "interactive" modes of access to the system are very similar, the programmatic interface consisting of FmpRunProgram calls that are functionally equivalent to entering the runstrings at a CI prompt. As an aid to programmatic usage, most of the programs return error status to the scheduling program and allow any output generated to be redirected to a file for programmatic processing.

Each printing task that the LP spooler handles is called a *request.* A request specifies that a certain collection of files is to be printed on a certain printer (or on any member of a *class* of printers, as discussed later) in a certain format. There are other properties associated with a request as well. For example, each request has a priority value that the spooler uses to determine which request should next be sent to a printer. Requests are usually created when a user runs the program named **lp**. Each request is assigned a unique *request ID* that is used when performing subsequent LP spooler commands on the request, and in tracking the progress of the request through the system.

The LP spooler must be told about each printer it may access. A name is associated with the printer. That name, rather than the device LU number, is used to refer to the printer within the LP spool system.

Output Destinations

The LP spool system refers to any device that it can output to as a *printer*, in keeping with the HP-UX terminology. The actual type of output device is not restricted by the LP spool system, although HP provides support only for printer and plotter devices. In this paper, the term "printer" is used to mean any output device, unless printer devices are specifically indicated.

Three general categories of printers are accessed by the LP spool system: *local printers*, which are actual print devices directly connected to the local system, *network peripherals*, which are printers connected to a LAN (not to any one system), and *remote printers*, which are network links to printers located on remote hosts.

Local Printers

Local printers are physical print devices directly connected to the local system, usually through a MUX, ASIC, or HP-IB interface. The LP spooler prints to any supported printer or plotter in a supported configuration using the standard RTE drivers.

Additionally, you can develop your own custom device handlers for other devices to which you wish to spool output. The FORTRAN and Macro source code to HP's local printer handlers is provided with the VC+ product, complete with instructions on how to modify the sources to support other devices.

Network Peripherals

The LP spooler prints to network peripherals connected to Ethernet LANs by HP JetDirect cards. You must have the NS-ARPA/1000 or ARPA/1000 subsystem installed to print to these devices. Network peripherals function somewhat as if they are local devices connected to your system via a LAN.

The initial release of the spooler does not support printers connected to a Data communications and Terminal Controller (DTC). However, the protocol used to access JetDirect network peripherals may also be used to access printers on certain DTC types. The JetDirect protocol accesses the printer by opening a TCP connection to a certain TCP port on the IP address of the network peripheral, sending the data to be printed across the connection, and closing the connection. If you have a DTC printer that may be accessed in an identical fashion, but using a different TCP port number, then the LP spooler may be configured to allow the printer to be accessed. The LP spooler network peripheral printer interface allows the destination TCP port to be configured for just such a purpose. Note that DTCs from some manufacturers do not flush any buffered data received from the network when the connection is closed. Thus, the DTC printer handler must send enough null characters (ASCII 0) to flush out any data buffered in the DTC before closing the connection. This author has no experience using DTCs, and so cannot provide pointers on which models behave in this manner.

Remote Printers

A *remote printer* is a network link to a printer connected on a remote host. A request sent to a remote printer is transferred into the spool system on the remote host via the NS-ARPA/1000 or ARPA/1000 subsystem. The transfer is accomplished using a semi-standardized Internet protocol. The protocol also supports performing these operations on the local spool system from a remote host:

- Cancelling print requests.
- Obtaining listings of the spool system queues.

In this manner, each host in your network may access any printer connected to any other host in the network. The host types that support this network printing include other RTE

Interface	Usage
`generic`	Generic non-PCL local printer
`pcl`	Local PCL printer
`passthru`	Local plotter or other non-RTE-printer device
`hpnp_pcl`	PCL network peripherals
`hpnp_passthru`	Network peripheral plotters
`rrte`	Remote printers on RTE hosts
`rhpux`	Remote printers on HP-UX hosts
`rbsd`	Remote printers on BSD UNIX hosts

Table 1: LP spooler printer interfaces.

hosts running NS-ARPA/1000 or ARPA/1000, HP 9000 hosts running HP-UX, and BSD UNIX machines to which NS or ARPA connectivity from the HP 1000 is supported. Both clients and servers for the network printing protocol are provided with the LP spooler. Thus, network spooling may occur in either direction between RTE and UNIX machines, and between RTE and other RTE-A/VC+ machines.

Printer Interfaces

Device-specific print handlers, known as *printer interfaces*, may be defined for each kind of printer. In general, a printer interface is a set of routines linked into a program named **lpout**. Each printer interface has a name that identifies that interface to the **lpout** program.

HP supplies a number of printer interfaces with the LP spooler. A list of these interfaces is shown in Table 1. In that table, the term "PCL" refers the Printer Control Language implemented by most HP printers. No LP spooler printer interface provides printer configuration options for PostScript[3] printers, but PostScript-formatted documents may be spooled to capable printers by sending the data in "raw" format to the printer.

As mentioned previously, the source code to the local printer interfaces is included with the LP spooler. You may add your own interfaces or customize the existing ones, following instructions included in the source code. The portion of the PCL local printer interface that handles PCL-related options is also used by the PCL network peripheral printer interface. Hence, changes made to the PCL handler for local printers may also be loaded into the PCL network peripheral interface.

Printer Classes

A *class* of printers may be defined in the LP spool system. A printer class is a named collection of local printers of similar type. For example, if a host has two printers connected, a LaserJet Plus and a LaserJet III, then class "laserjet" could be created with both printers as members of the class. If a print request names a class, rather than a specific printer, as the desired destination then the request is printed on the first available member of the class. Using the same example, specifying destination "laserjet" in a request prints the request on either the LaserJet Plus or the LaserJet III, depending on which printer first becomes available. A printer may be a member of only one class. Neither remote printers nor network peripherals may be made members of a class.

---

[3] PostScript is a registered trademark of Adobe Systems Inc. in the U.S. and other countries.

Spooled File Formats

The LP spool system does not create files to be printed, as can the SP spool system when spooling output from an LU to a spoolfile. Instead, the LP spooler prints files that have already been created by some other means.

Any RTE file type may be spooled. In general, the files fall into three categories:

- Record-structured text files of type 0 or types 2 and above. These files are printed as a series of FMP records, using a separate XLUEX call to write each record if printed on an RTE system.
- Redirected LU output captured using the SP spool system. These files may have headers that contain the EXEC CNTWD parameters and EXEC(3) control request information from the redirected EXEC calls that generated the output[4]. The LP spooler can use these headers when printing the file on an RTE system to reproduce the print formatting specified in the original EXEC calls.
- Files of type 1, which contain a UNIX-style "byte stream" of print data that is not organized into FMP records. Type-1 files must conform to a special format used by the LP spooler. These files are normally introduced into the spool system only by the LP spool system itself when receiving a request spooled from a remote UNIX host.

Request Options

Various options may be specified in a request to specify print formatting and other request characteristics. Many options are interpreted by the printer interface that handles the printing of the request, thus, the documentation for the destination printer interface must be consulted to determine the set of options available. The following is a generalized list of the types of options that may be entered for a request that is printed using an RTE printer interface. Note that some of these options are standard System V spooler options, and others are interface-specific options that may not be interpreted by all RTE printer interfaces. Options are available that set the following request characteristics (among others):

- Set the number of copies to print.
- Set the priority of the request.

---

[4] The feature of retaining EXEC headers in user-specified spool files is new in the 6.1 release of the SP spooler.

Program	Usage
lp	Create a print request
lpstat	Display spooler status
cancel	Cancel requests
lpalt	Alter requests
lpadmin	Spooler administration
lpsched	Start up spooler
lpshut	Shut down spooler
enable	Enable printers
disable	Disable printers
accept	Accept new requests for destinations
reject	Reject new requests for destinations
lpmove	Move requests between destinations
lpfence	Set printer priority fence
lpout	Output to directly-connected local printer
rlpout	Output to remote printer or network peripheral
rlpdaemon	Remote printing daemon for incoming requests

Table 2: LP spooler programs.

- Send mail or run Notify when the request finishes printing.
- Add a user-defined banner to the banner page.
- Inhibit printing the banner page.
- Treat column one of the print data as FORTRAN-style carriage control.
- Inhibit standard driver-processing of the print data (such as treating trailing underscores as line continuation, etc.).
- Suspend before and after output for forms changes.
- Set various PCL characteristics (such as landscape mode, compressed pitch, etc.).

The LP Spool System Programs

Although this paper does not delve into the details of LP spooler usage, a listing of the programs provided may be useful to those of you who are familiar with the System V spooler. The LP spool system includes an array of programs that may be overwhelming at first. The programs are listed in Table 2 together with a brief description.

HP-UX Compatibility

The reader familiar with the HP-UX spooler may be curious as to how much of the HP-UX offering is implemented on RTE, as well as other compatibility issues.

The RTE LP spooler provides very much the same functionality as its HP-UX counterpart, especially from the point of view of a user (as opposed to a system administrator). In general, the runstring syntax of each RTE program is almost identical to its HP-UX counterpart, within the limits imposed by the different runstring handling philosophies of the two systems. The RTE implementation of some functionality is substantially different than the HP-UX implementation in an attempt to provide greater efficiency within the RTE environment. This is why two programs appear in Table 2, **lpout** and **rlpout**, that do not have UNIX counterparts, and certain "lower-level" HP-UX programs are absent, such as **rlp** and **rcancel.**

Of the spooler "proper", the one utility missing is the **lpana** program, which prints spooler performance analysis information. This program is apparently not in widespread use; many people with extensive HP-UX spooler experience have never run it.

Certain **lpadmin** options related to the UNIX I/O system and the HP-UX Diskless Clustered Environment are not provided. No provision exists for specifying custom remote printer cancel and status interrogation procedures; these functions are "hard-coded" to operate using the standard protocols.

The RTE implementation provides most of the HP-UX print-time formatting options that simply specify PCL escape sequences to be sent to printers. In fact, we added certain options that we felt to be of value but which do not appear in the PCL-related model scripts on HP-UX at present. However, RTE does not provide all the printer formatting utilities that HP-UX does. This includes formatting utilities that are usually run before printing, such as **pr** and **asa**, and output filters typically invoked by model scripts, such as **lprpp, divpage, reverse**, etc. Thus, HP-UX offers a greater wealth of print formatting styles than does RTE. However, some of these utilities may be provided for RTE in the future, as customers inform HP of missing print formatting functionality that is important to them.

Differences in the I/O systems between RTE and HP-UX may cause some (hopefully minor) headaches. Problems may arise due to printer output processing that is performed at the driver level on one system but not on the other. For example, the treatment of tab characters and FORTRAN-style column one carriage control is different between the two systems. This is another area where customer feedback can be useful in letting HP know what problems are encountered in the real world, and how the RTE LP spooler can be enhanced to coexist more peacefully in customers' networks.

The implementation of printer interfaces on RTE is substantially different than that of HP-UX. On HP-UX an interface usually consists of a shell script that is invoked to perform printer output. We chose not to go with that approach on RTE largely because the limited CI command language does not lend itself to implementing this relatively complicated task. The amount of code necessary to perform this task is also typically higher on RTE than on HP-UX for a number of reasons related to the differing process scheduling and I/O systems.

On HP-UX, a file that has been spooled for printing in a request may safely be purged, even if the request has not yet been printed. Not so on RTE, unless the spooler is specifically told to make a copy of the spooled file somewhere else at request creation. See the following section on *Symbolic Link Files* for more information.

RTE Implementation Discussion

The remainder of this paper discusses various LP spooler implementation details that may possibly be of interest, primarily to RTE programmers.

Disk Usage

The LP spooler keeps almost all of its state information on the disk. The spooler status is contained in various files in a layout similar to that of HP-UX, although the contents of each file are in an RTE-specific format. We won't delve into the details of these files here; suffice to say they hold information about queued requests, the status of printers and printer classes, including which request is currently active on which printer, and so forth. The spooler is thus a rather I/O-intensive subsystem, even apart from the actual output of print data. However, this scheme provides a number of benefits, including:

- Very little state is lost when the system is rebooted (this is a major complaint about the SP spooler). Any requests that were actively printing at the time of reboot will be reprinted in their entirety when the spooler is restarted, however. No method of restarting output at any point other than the beginning of the print data exists that is reliable for all print data and printer types.
- No "monitor" process must be present to coordinate spooler operations, as is the case for many RTE applications.
- Spooling activity is not restricted by the amount of information that can be held in a static-sized memory area, as would be the case if a memory-based approach were used.
- The inner workings of the spooler are rather visible to the outside world using the standard file system utilities. Each control file is human-readable, that is, the contents are represented as ASCII text. Accordingly, if any portion of the spooler "locks up" (heaven forbid) then the spooler need not be relied upon to obtain information on the state of the spooler, or to modify spooler "tables". Admittedly, it is a lot harder to do this yourself than to let the spooler handle the details.

One note for the future: the current industry trend on UNIX is to place the spooler files in directory **/var/spool/lp** instead of **/usr/spool/lp**, in an attempt to separate "static" portions of the file system from more "dynamic" portions. The RTE implementation may eventually switch to that directory structure for compatibility.

Lock Files

Since competing processes are reading and updating information at the same time without a central controlling process present, a means of coordinating access to those files is needed. The file "open flags" maintained by the file system are useful in this regard, but alone are insufficient for the needs of the spooler. The spooler updates text files of variable record lengths, requiring an existing file to be copied to a new file that incorporates the changes. If this operation is to be performed with the existing file open exclusively (as the means of "locking" the file to the updating process) then the new file must then be copied back on top of the existing file before the existing file is closed. This results in an extra shuffling of the data on the disk that seems wasteful.

Access to files that are updated by competing processes is coordinated through the use of *lock files*. Lock files are a concept more familiar on UNIX systems, which historically have not implemented exclusive file opens[5]. In general, the presence of a lock file on the disk tells cooperating programs that some process has an associated data file or files "locked" to it. A process that successfully creates a lock file is granted sole access to some other set of disk-resident information, as defined by the cooperating processes. When access is complete then the lock file is removed. In this manner, the file system is used to provide a semaphoring technique.

The usage of lock files in the RTE spooler does rely on the exclusive file open feature of the file system for reliability. A program that requires access to a data file that is controlled in this manner attempts to exclusively open a lock file that has the same path and name as the data file, but which has type extension **.lock**. If the lock file is already open then the program sleeps for a few seconds and then retries opening the lock file. When the lock file is opened successfully then the program holds exclusive access to the data file. The reliability provided by the exclusive open flags that we mentioned above comes into play when a locking process is aborted for some reason, since the exclusive open is released by the file system.

Symbolic Link Files

The LP spooler is perhaps the first HP-supplied application to make use of the *symbolic link files* capability added to the file system at the 6.0 release[6]. The spooler uses these files to avoid file copying overhead and disk space usage by the print data files in requests.

Both the HP-UX and RTE spoolers do not, by default, make their own copies of the files to be printed (unless the file is transferred to a remote system for printing). Instead, the print data files are left where they are and links to the files are made in directory **/usr/spool/lp/request/***printername*. All access to the data files is made through these link files.

On HP-UX, a different kind of link file known as a *hard link* is made. We won't try to explain these in any depth here; suffice to say, this type of link functions as an alternate directory entry for the file linked to. If the original file is purged then the disk blocks containing the file data are not actually reclaimed until the hard link is also purged. Therefore, if a data file is purged before the request is printed then the operation of the spooler is not disturbed – the file is still printed. RTE does not contain the hard link feature in its file system; we only have symbolic links available. Symbolic links provide a much looser coupling with the linked-to file that does not prevent the original file from being purged in its entirety. Thus, the RTE spooler cannot print any data file that is purged beforehand. If you wish to do this then you must instruct the spooler to make a copy of the file, rather than a symbolic link to it.

---

[5] An even more flexible feature known as *record locking* is now prevalent on UNIX systems.

[6] See the paper *Symbolic Links on RTE-A/VC PLUS* by Gary Gan, 1992 INTEREX Proceedings paper number 1010, for more information.

Network Spooling

The network spooling implementation is based on the LPDP protocol documented in Internet document RFC-1179[7]. This protocol has not been formally standardized, but has become something of a *de facto* standard, and is likely to remain reasonably stable[8].

The networking-specific code of the RTE implementation of LPDP clients is written in the C language using the BSD IPC interface. This interface is handy for writing applications that interact with UNIX, since the standard behavior of the calls is so similar to that of UNIX. Unfortunately, the networking code of the RTE servers is written using the proprietary NetIPC interface, since the **inetd** program is used to listen for connections from clients; **inetd** at present cannot listen for connections that are serviced by BSD IPC programs.

UNIX Spooled File Format

As mentioned in a previous section, data files spooled from UNIX are represented as type-1 files that contain the unaltered byte stream of the UNIX file. This special format allows the spooler to accomodate any UNIX file without truncating data.

The reader familiar with the UNIX file system will recall that UNIX text files are stored as a series of characters with line-feed ("newline") characters separating lines. The LP spooler does not attempt to collect newline-terminated lines of data files spooled from UNIX into separate FMP records (which would require truncating lines at an arbitrary maximum buffer size). There is no particular need to do so, since the spooled data file will be handled solely by the spooler; it does not need to be formatted for processing by the standard set of RTE record-oriented utilities, such as **grep**, **li**, etc. In addition, the spooler cannot know at the time a data file is transferred into the system whether the file should be processed as ASCII text or as binary data. HP deemed unacceptable any file representation scheme that would allow binary data to be lost, such as a scheme that would drop bytes that do not fit within an FMP record assembly buffer. Accordingly, the data of UNIX files is stored without format conversion. Type-1 files were chosen to hold these files, since no FMP record unpackaging overhead is incurred for that file type.

The LP spooler imposes a certain structure on the type-1 files it creates and processes. The first 12 characters of the first block of a type-1 file must contain an ASCII representation of the number of valid bytes in the file. This count tells the spooler how many bytes to actually print, since type-1 files contain no EOF position information. The rest of block 1 is unused. The remaining blocks, starting at block 2, contain the byte stream to print. The directory entry for the file would make a much handier place to store the valid byte count than does the first file data block. We avoided redefining any of the existing file size fields of the directory entry to hold this count because we didn't want to break any existing FMP routines or utilities. We didn't stash the byte size in some other relatively harmless directory entry field, such as the "create time" field, because the bogus value would cause some FMP utilities, such as **dl**, to report nonsense that might alarm some customers. It is possible that an FMP-based solution to this problem will be implemented in the future, as HP continues to focus efforts on UNIX interoperability issues.

---

[7] McLaughlin, L. III. *RFC-1179: Line Printer Daemon Protocol.* 1990. Available from the Internet Network Information Center at the Stanford Research Institute in Palo Alto, CA.

[8] RFC-1179 does not propose a standard; it is published for informational purposes only. The status of this protocol as registered by the Internet Activities Board (IAB) is "informational", indicating it is not planned to be adopted as an Internet standard.

Note that the spooler *does* have to convert files from FMP record format to UNIX text format when transferring files from RTE to UNIX (and the request options do not indicate that the files are binary data to be transferred verbatim). This conversion basically entails adding a newline character to the end of the data of each record sent. Spooling files to UNIX is an even higher overhead operation than you might expect, because the remote spooling protocol dictates that an ASCII representation of the number of bytes in a file be sent across the connection first, followed by that number of bytes of file data. The spooler must first read through the entire file, counting data bytes and adding one byte for each record to account for the "newline" character that will be appended. After sending the byte count to the remote server, another pass through the file is made to send the file data. RTE does keep file size information in the EOF position field of the directory entry, but the size is a word, not byte, count that includes the overhead words that hold the length of each record.

The RTE LPDP Protocol Extension

The RTE implementation of the LPDP protocol includes a minor extension that is used when spooling requests between two RTE systems. This extension saves considerable overhead by not performing the data format conversions that are required when spooling files to UNIX (including the byte counting pass through the file). It is because of this difference in spooling to RTE systems versus UNIX systems that separate interfaces are provided for remote spooling to the two system types.

The RTE implementation adds a new command code to the standard protocol. Non-RTE systems reject the additional command code, such that an attempt to treat a remote UNIX host as if it were an RTE host fails. This code enables the two RTE machines to transfer the file in "forced type-1" mode[9], where the file data sent consists of the unaltered 128-word blocks read from the disk. Thus, FMP record unpacking overhead is also avoided.

The new command also contains information that allows the destination RTE host to reconstruct the proper file type (for correct interpretation of the file's record structure), EOF position, and record count (for cosmetic reasons). This method of file transfer is similar to the processing normally performed by the FmpCopy routine, which copies files in "forced type-1" mode and sets the destination file attributes to those of the source file.

---

[9] This name comes from the description of the "F" option of the FmpOpen call, which specifies this mode of file access.

Note that the spooler preserves the original file attributes on the destination system by necessity arising from the "forced type-1" mode file transfer. One naturally expects any RTE application that transfers files between RTE systems to ensure that the destination file is identical to the source file in these respects. But the basic goal of the spooler is not to provide a generalized file copying service, but to print files. Is the spooler required to preserve the original file attributes on the destination system in order to generate the correct output? Let us suppose the spooler did not implement the protocol extension discussed here. In that case, a remote spooling transaction would simply convert any RTE record-structured file into UNIX text format during the transfer (without knowing whether the destination host was an RTE or UNIX system). Could the spooler have been implemented such that the remote RTE system would reproduce the correct output from the resultant type-1 file, regardless of the original attributes of the spooled file?[10] In most cases, the answer is "yes". But there is one case where the conversion from RTE to UNIX text format is an "information losing transaction": the case of a record-structured file that contains a newline character (line-feed, ASCII code 10) inside a record. Thus, while the extended protocol effectively avoids this problem for RTE-to-RTE transfers, it remains a problem for RTE-to-UNIX spooling. This problem perpetually causes headaches for programmers that implement RTE-to-UNIX text file transfers. Fortunately for the spooler, newline characters tend to be embedded only in "raw" (that is, binary) print images that are transferred to UNIX verbatim, without conversion to text file format. The spooler may be told which format, text or binary, to use in sending the file to the UNIX system. This is similar to the usage of the "ASCII" versus "binary" data transfer modes of the FTP utility.

Just How Good Is This Network Spooling, Anyhow?

Let's be up front about the quality of the System V network spooling implementation: it isn't very good. The LPDP protocol is reasonably well suited to the basic task of shipping a request to a remote machine for printing – when no errors are encountered along the way. Thus, the protocol does get the job done most of the time, and many users don't have major complaints about the network printing setup. But there is plenty of room for improvement on this protocol.

The protocol contains precious little support for status to be sent from spooling servers to clients, such as to inform the client of errors encountered during the processing of an operation, or even that the operation was successful. Because of this, it is often difficult for the user to determine whether a remote operation was executed as intended, and if not, why not. For example, the "remote cancellation of print requests" operation returns no status at all. This is why the **cancel** program is somewhat taciturn while cancelling remote requests: **cancel** has no idea what was accomplished on the server side. For other operations a simple "pass/fail" status is returned, providing only tepid friendliness at best.

The LPDP protocol requires the use of 8-character host names, an archaic form of host naming kept alive by this and other antiquated UNIX network services. Part of the protocol for transferring a request to another system involves transmitting file names that contain the name of the originating host in the file name. These file names must be acceptable to UNIX implementations with a maximum file name size of 14 characters. Accordingly, we can no longer avoid the need for this host naming convention on the HP 1000. At present, the 8-character host name is kept in a file specific to the spooler. If other future services also need access to this host name then it may have to be moved to a more generally-accessible location.

---

[10] Note that the spooler was *not* implemented in this fashion; therefore, the **rrte** interface should be used for remote access to RTE hosts.

Some of the "clunkiness" of the networking implementation results from the grafting of the BSD protocols onto the System V spooler, which was not originally intended to be used in any networked fashion. One outgrowth of this situation is that the networked usage and behavior of many of the spooler programs is not exactly obvious. Certain programs do not function remotely at all. This presents a documentation challenge that is seldom, if ever, met to everyone's satisfaction.

Another element of the System V spooler that suffers from the introduction of networking is the *request ID* syntax. Recall that a request ID is a unique identifier for a particular request assigned by the spooler. For various reasons, some of which we won't bother to go into here, a request ID must often be modified to a new value when a request is transferred to another system. Not only does this diminish the usefulness of the request ID in tracking the progress of a request through the network, but in certain cases the originating user may find it difficult to determine the new value of the request ID, complicating remote operations.

In the RTE implementation, we chose to make the name of the host on which the request originated an integral part of the request ID. For any request transferred into an RTE system, the request ID has the string "*@hostname*" tacked onto it. This is a rather significant diversion from HP-UX functionality, which we generally sought to copy almost exactly, but it provides these two important benefits:

- A unique request ID generated on one machine never "collides" with a request ID that was generated on another machine. If the originating host name is not included in the request ID then two requests may arrive on the same host that have the same ID. If ambiguity in request IDs is to be avoided then one of the IDs must be modified to preserve uniqueness, thereby complicating matters for the originating user who no longer knows how to refer to the request.
- The originating host is always identified whenever the request is referenced in some manner, for example, in a queue status listing or log file entry. This information is felt to be important enough to include in each mention of the request.

In general, the networking implementation is functional but inadequate by today's standards for networked services. More modern spool systems not only correct many of the deficiencies listed here, but are implemented fully under the client-server paradigm. Each component of the system is intended to be used within a network; much of the spooler "state" information kept privately on the local host in the System V spooler is made available to the entire network. This allows, for example, "classes" of similar printers to be distributed across the network, such that a request may be printed on the first available printer on any machine in your network (with some sort of notification to the user as to where the printed output may be found).

To Conclude

This paper has presented HP's long-awaited solution to the spooling woes of RTE-A. We at HP hope that you find the LP spool system to be a solution that is powerful, flexible, and easy to integrate into your network.

The HP-RT Real-Time Operating System

Paper #1008

by Kevin D. Morgan

Hewlett-Packard Company
11000 Wolfe Road MS-42UN
Cupertino, CA 95014-9804
(408)-447-5079

© 1993 Hewlett-Packard Company, reprinted with permission.

An operating system that is compatible with the HP-UX operating system through compliance with the POSIX industry standards uses a multi-threaded kernel and other mechanisms to provide guaranteed real-time response to high-priority operations.*

HP-RT† is Hewlett-Packard's real-time operating system for PA-RISC computers. It is a run-time-oriented product (as opposed to a program-development-oriented product) based on industry standard software and hardware interfaces. HP-RT is intended to be used as a real-time data acquisition and system control operating system. It is designed around the real-time system principles of determinism (predictable behavior), responsiveness, user control, reliability, and fail-soft operation. These characteristics distinguish a real-time operating system from a nonreal-time operating system. This article reviews some of these characteristics of HP-RT and discusses the specific designs used to provide these features.

HP-RT runs on the HP 9000 Model 742rt VMEbus board-level computer, which is based on HP's PA-RISC 7100 technology . The 742rt is designed to fit into a VMEbus card cage or an HP 9000 Model 747i industrial workstation cabinet.[1]

The HP-RT kernel is compatible with the HP-UX operating system through compliance with the following industry standards:

- POSIX (Portable Operating System Interface) 1003.1, which defines a standard set of programmatic interfaces for basic operating system facilities
- POSIX 1003.4 draft 9, which defines the standards for real-time extensions
- POSIX 1003.4a draft 4, which defines the standards for process-level threads.

HP-RT also supports C/ANSI C, C++, PA-RISC assembly language, and many SVID/BSD (System V Interface Definition/Berkeley Software Distribution) commands and functions.

HP-RT Software

The HP-RT software is divided into two main categories: the HP-RT kernel and the optional HP-RT services (see Fig. 1).

HP-RT Services

The optional HP-RT services include the following components:

---

† HP-RT is derived from a third-party operating system called LynxOS from Lynx Real-Time Systems Inc. All kernel-level algorithms and data structures described in this paper are based on LynxOS features.

- Network services including the Network File System (NFS), TCP/IP, Berkeley sockets, and ARPA/Berkeley networking services
- Libraries for developing OSF/Motif graphical user interfaces and X clients
- Development tools to help users create applications to run in the HP-RT environment
- Cross debuggers hosted on an HP-UX development workstation for debugging the HP-RT kernel or applications running on an HP-RT target system.

Kernel Software

The HP-RT kernel is designed so that it can be scaled to balance memory and performance requirements. It is small to reduce overhead. The kernel components include:

- A counting semaphore mechanism for process synchronization and to help ensure atomicity around critical sections of code.

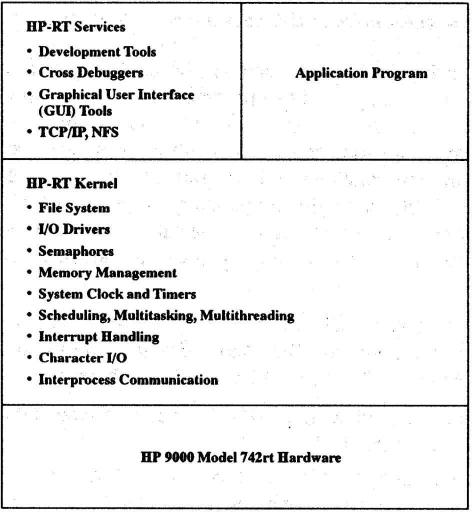

HP-RT Services
• Development Tools
• Cross Debuggers
• Graphical User Interface (GUI) Tools
• TCP/IP, NFS
Application Program
HP-RT Kernel
• File System
• I/O Drivers
• Semaphores
• Memory Management
• System Clock and Timers
• Scheduling, Multitasking, Multithreading
• Interrupt Handling
• Character I/O
• Interprocess Communication
HP 9000 Model 742rt Hardware

Fig. 1. The HP-RT kernel and services.

- A system clock that generates time interrupts every 10 milliseconds. Thus, time events using standard software interfaces have a 10-millisecond resolution. For higher timing accuracies, drivers and user processes can access the hardware timers on the Model 742rt. These timers have 1-µs resolutions and are 16 and 32 bits wide.
- I/O drivers for Ethernet, SCSI II, RS-232-C, and parallel I/O for the Model 742rt computer, and guidelines for writing VMEbus drivers
- Standard operating system services such as:
- Scheduling, multitasking, and multithreading
- Memory management
- Interrupt handling
- Character I/O

- Interprocess communication
- POSIX 1003.1, .4, and .4a kernel services.

Many of these components are described in more detail later in this article.

HP-RT Development Environment

The development environment for HP-RT is shown in Fig. 2. Programs created to run on the Model 742rt in the HP-RT environment are developed (using PA-RISC compilers and linkers) on an HP 9000 Series 700 or 800 HP-UX system. The executable programs can be downloaded via LAN to a local disk on the target system (Model 742rt), or implicitly downloaded when the program is executed via NFS mounting between the HP-RT and HP-UX systems. The user can debug the downloaded program from the host system via the RS-232-C and LAN connections between the two systems. Users can customize the SoftBench software development environment[2] on the development host to launch programs to a remote HP-RT system and to launch the correct program debugger for HP-RT program debugging.

The items that come with the HP-RT development toolkit include:

- Libraries for building HP-RT kernels and user programs

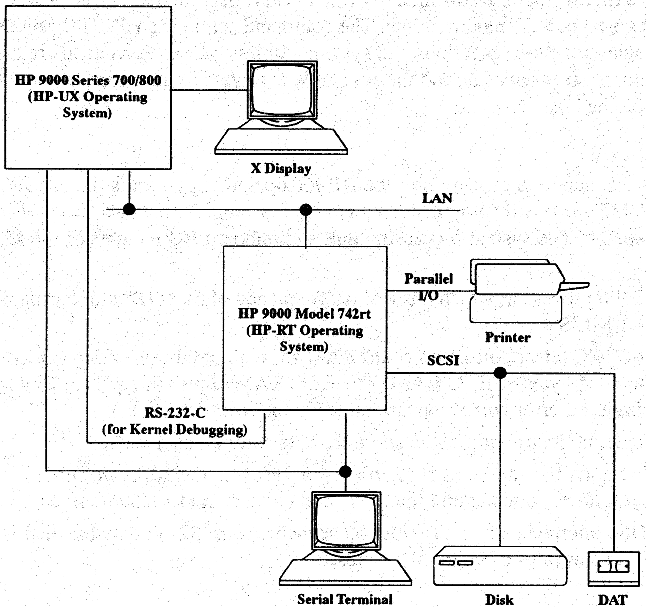

HP 9000 Series 700/800 (HP-UX Operating System)
X Display
LAN
Parallel I/O
Printer
HP 9000 Model 742rt (HP-RT Operating System)
SCSI
RS-232-C (for Kernel Debugging)
Serial Terminal
Disk
DAT

Fig. 2. The HP-RT development environment.

- Include files for compiling user programs and I/O drivers for executing in an HP-RT operating environment
- Installation and user program compilation scripts
- A pair of source-level debuggers: one for user program debugging and one for I/O driver and kernel-level debugging.

The two remote debuggers included with the HP-RT development kit are derived from the standard *xdb* debugger product provided with the HP-UX operating system. The debugger used for user program debugging is capable of debugging multithreaded user processes and communicating with the target HP-RT system using a TCP (Transmission Control Protocol) virtual circuit socket. The kernel debugger is for kernel-level and I/O driver debugging and communicates with the target HP-RT system via a dedicated RS-232-C serial communication link. Using a dedicated communication link allows the kernel debugger to operate without interfering with the normal operation of the target operating system.

A set of user commands, a bootable kernel, and miscellaneous files are included with the HP-RT system. These items can be installed via LAN on a disk connected to the target system. The HP-RT kernel can also be booted across a LAN and commands and user programs can either reside in RAM memory (via a RAM disk facility) or be accessed across the network via NFS mount points. The command set on the HP-RT target system is oriented around run-time operations and system administration. Commands related to program development (such as *cc* and the *rcs* and *sccs* tools) are not supported and can only be used on the host.

HP-RT Hardware

The hardware that supports execution of the HP-RT operating system is the HP 9000 Model 742rt VMEbus board computer. This system consumes consumes two slots of a VMEbus backplane. The system processing unit and onboard I/O features of the Model 742rt include:

- PA-RISC 7100 processor, which has a clock frequency of 50 MHz and is capable of executing 61 MIPS
- 8M bytes of ECC (error correction code) RAM for main memory, which can be upgraded to 64M bytes of ECC RAM (The ECC RAM comes in a pair of SIMMs and provides single-bit error correction and multiple-bit error detection.)
- 64K-byte external instruction cache and 64K-byte external data cache
- Onboard I/O ports for one SCSI II interface (up to seven devices), two serial RS-232-C interfaces, one parallel interface, and one Ethernet LAN interface
- VMEbus D64 interface, which provides an asynchronous, 32-bit data bus that is capable of transfer rates of up to 40 Mbytes/s.

The Real-Time Kernel

The HP-RT kernel and I/O drivers are designed for real-time response and determinism at a level never before accomplished in a Hewlett-Packard operating system product. The HP-RT kernel ensures that the highest-priority operations are serviced within 50 to 110 microseconds in the worst case and typically much faster depending on the specific operation. To accomplish this, the HP-RT kernel uses a fully reentrant and interruptable design and makes extensive use of full kernel support for threads for user and kernel processes.

Multithreaded Kernel

The fundamental unit of an executing task in HP-RT is the concept and structure of a thread. A thread contains a program counter (next instruction pointer) and a stack for recording local subroutine variables and calling sequence parameters. Threads do not own a specific address space or a specific set of code. Threads typically share address space (data area) and code with other threads. The concept of a process is simply a combination of a single thread, a code segment, and a data area (see Fig. 3a). HP-RT extends this concept by allowing a single process to create multiple threads (see Fig. 3b). These additional threads execute code in the same process code area and have identical access rights to all data areas in the process. See "An Overview of Threads," on page 1008-8 for a brief tutorial on threads.

HP-RT also implements the concept of a kernel thread. A kernel thread is a thread of execution that only executes kernel code at a kernel privilege level. Kernel threads are used in HP-RT to provide kernel services asynchronously for any specific user process or thread with each service executing at a user-specified priority.

Reentrancy and Interruptability

The HP-RT kernel's general model is to execute on behalf of a thread of execution with interrupts enabled and context switching allowed. The specific thread executing may be a thread associated with a user process or a kernel thread. All threads, regardless of type, have their own user-specified priority, scheduling policy (time-sliced versus run-to-completion), and system level.

The system level is a specification of the mode in which a thread is executing. At system level zero, a thread runs in user mode, with user-level privileges. Kernel threads by definition never use this system level. At system level one, a thread executes kernel code with kernel-level privileges and with all interrupts enabled and context switching allowed. At system level two, a thread executes kernel code with context switching disabled, but interrupts enabled. Finally, at system level three, a thread executes kernel-level code with both context switching and interrupts disabled. Table I summarizes these system levels and execution modes.

Context switching and interrupt handling in HP-RT are described in more detail in paper 1010.

The HP-RT system supports one nonthread mode of execution, which is based on execution using a single interrupt stack. However, unlike timesharing systems and many real-time systems, HP-RT makes very limited use of interrupt-stack-based execution because this mode of execution is always at a higher priority than thread execution. Execution using an interrupt stack means that a full thread context is not established, which means that a context switch to a thread cannot be allowed until the interrupt-stack-based execution is complete. Most interrupt service routines, such as the handlers for the SCSI bus and LAN interrupts, are instead handled by a specific kernel thread. These threads are scheduled when their corresponding interrupt occurs at their specific priority and are not executed until all higher-priority thread execution is complete.

Table 1. System Levels and Execution Modes

System Level	Execution Mode	Context Switching	Interrupts
Zero	User	Allowed	Enabled
One	Kernel	Allowed	Enabled
Two	Kernel	Disallowed	Enabled
Three	Kernel	Disallowed	Disabled

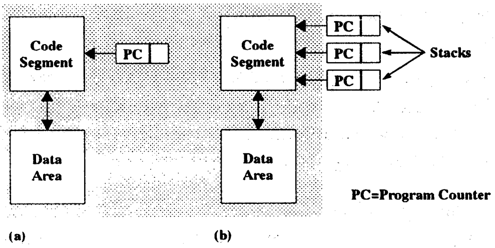


Fig. 3. Thread configurations. (a) A typical single-thread process. (b) A multiple-thread process.

Because of the general reentrancy of HP-RT, explicit calls are used in kernel code and I/O drivers for managing reentrancy.† The macros *sdisable()*, *srestore()*, *disable()*, and *restore()* are used to move a process to system levels two (context switch disabled) or three (both context switching and interrupts disabled) and back to the premove system level. Turning context switching off guarantees atomicity with respect to the execution of other threads. Turning off interrupts guarantees atomicity with respect to execution of both threads and interrupt-stack-based handlers.

Data structures used by the kernel are generally global to the entire kernel and nonreentrant operations must be properly protected. A simple example of this is the *use_count* field of the in-core *inode*†† data structure. The *use_count* field indicates the number of instances of a particular file that are active (e.g., *open*). When a new process accesses an *inode*, the equivalent of the code statement *inode_ptr->in_use++* (increment *use_count*) must be executed. On PA-RISC (and most RISC processors), this code translates to a sequence of instructions that loads the *use_count* value, increments it, and then stores the value to the memory location it came from. Interleaving such operations, which can easily happen because of a context switch from one thread to another, will cause the *use_count* to miss an increment, producing devastating long-term results.

† A reentrant process consists of logically separate code and data segments and a private stack. Multiple instances of a reentrant process can share the same code segment but each instance has its own data segment and stack.

††An *inode* is the internal representation of a file in a UNIX*-system-based operating system. An in-core *inode* is one that resides in main memory.

For example, Fig. 4 shows what can happen when a thread is interrupted before finishing incrementing the *use_count* field for a particular *inode*. The *use_count* field is represented by the variable X, which is initially equal to one (i.e., some other thread or process is accessing the same file). At ⓐ Thread 1 begins executing the instructions to increment X, but just before storing the result in X, Thread 2 interrupts at ⓑ and the scheduler hands control over to Thread 2. Thread 2 increments the same *use_count* field. When Thread 2 is finished, X = 2 and the scheduler returns control back to Thread 1 at ⓒ. At ⓓ Thread 1 finishes its work on the *use_count* field by storing the value it computed before being interrupted into X. At this point X should be equal to three, but because Thread 1 was interrupted before it finished its critical section, X = 2.

The need for atomic increment and decrement operations is so pervasive in the HP-RT kernel that special macros called *ATOMIC_INC()* and *ATOMIC_DEC()* are used. These macros generate inline assembly code that disables interrupts, performs the increment or decrement operation, and reenables interrupts.

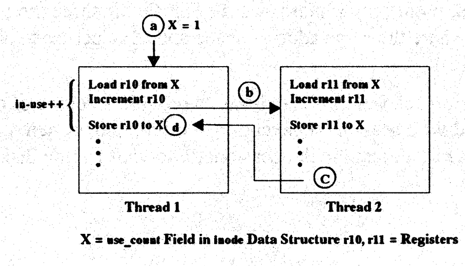


Fig. 4. What can happen when a thread is context switched in the middle of a critical operation. Thread 1 is interrupted and context switched just before it is about to increment the *use_count* value. As a result, when Thread 1 is finally able to finish its operation, the wrong value is stored in *use_count.*

Use of an interrupt disable versus a context switch disable is a key design decision for every critical section of HP-RT kernel code. The main question asked in arriving at a decision is whether the operation is critical relative to execution of code that can run on the interrupt stack. Since very little code in HP-RT executes on the interrupt stack, a context switch disable usually suffices for protection. However, a context switch disable is a more expensive operation than an interrupt disable operation. A context switch requires memory access and an interrupt disable only requires execution of an inline assembly statement which turns off the interrupt enable bit in the PA-RISC processor status word. Thus, very short operations are better protected with interrupt disables.

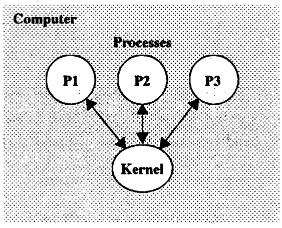
This raises the question of how HP-RT solves the problem of long critical sections for which a context switch or an interrupt disable last too long. In the analysis of customer requirements and competitive systems, it was determined that context switch off times should be held to as close to 100 microseconds as possible, and ideally less, and interrupt disables should be held as close to 50 microseconds as possible, and ideally less. Longer critical sections are managed using kernel-level semaphores.

An Overview of Threads

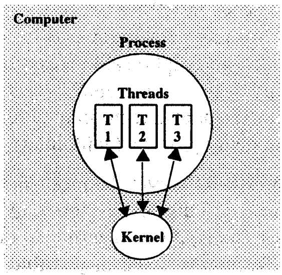
When a process is running it executes a sequence of instructions stored in its address space in memory. This execution of a sequence of instructions is called a thread of execution, or simply a thread. The execution of a thread requires that it have its own program counter to point to the next instruction in the sequence, some registers to hold variables, and a stack to keep track of local variables and procedure call information. Although threads have some of the same characteristics as a regular process, they are sometimes called a "lightweight" process because they don't carry around the overhead (or extra weight) of regular processes. Table 2 lists some typical items associated with each thread and each process.

Fig. 5 models processes and threads running in a computer. The processes in Fig. 5a have one thread of execution each. They also have their own address spaces making them independent of each other. To communicate with each other (for example, to share resources) they must do so through the system's interprocess communication primitives, such as semaphores, monitors, or messages. In Fig. 5b the three threads are in one process. Thus they share the same address space and have access to all the per-process items listed in Table 2.

One of the reasons threads were invented was to provide a degree of quasiparallel execution to be combined with sequential execution and blocking system calls. For example, consider a file server that must block occasionally to wait for the disk. In a single-process



(a)



(b)

Fig. 5. Models of processes and threads running in a computer. (a) Multiple processes. (b) Multiple threads in one process.

situation the server would get a request and service it to completion before moving on to the next request. Thus, no other requests would be serviced while the server is waiting on the disk. If the machine is a dedicated file server, the CPU is also idle while the server process is waiting on the disk.

Table 2. Items Associated with Threads and Processes

Per-Thread Items*	Per-Process Items
Program counter	Address space
Stack	Global variables
Registers	Files
	Child processes
	Signals
	Semaphores

* All per-thread items are also per-process items.

If the server is a multithreaded process, one thread could be responsible for reading and examining incoming requests and then passing the request to a thread that will do the work. When a thread must block waiting on the disk, the scheduling thread can get another request and invoke another thread to run. The result of using threads in this case would be higher throughput because the CPU would not sit idle, and better performance because it is much faster to switch threads than to switch processes.

In a real-time system where a quick response to interrupts and other events is critical, threads offer some definite advantages, especially if one considers context switching between processes versus switching between threads. Table 3 summarizes some of the main differences between threads and processes.

Table 3. Differences between Threads and Processes

Processes	Threads
Program-sized	Function-sized
Context switch may be slower	Context switch may be faster
Difficult to share data	Easy to share data
Owns resources such as files and memory	Owns stack space and registers only

Bibliography

1. T. Anderson, et al, "The Performance of Thread Management Alternative for Shared-Memory Multiprocessors," *IEEE Transactions on Computers,* Vol. 38, no. 12, December 1989, pp. 1631-1644.
2. A. S. Tanenbaum, *Modern Operating Systems,* Prentice-Hall, 1992, pp. 507-523.
3. P. Dasgupta, et al, "The Clouds Distributed Operating System," *IEEE Computer,* Vol. 24, no. 11, November 1991, pp. 34-44.
4. R. Lafore and P. Norton, *Peter Norton's Inside OS/2,* Simon & Schuster, Inc., 1988, pp. 134-174.

Kernel Semaphores and Priority Inheritance

An example of an extended critical section is the manipulation of an in-core *inode*. Critical *inode* operations such as the addition of a file to the directory data of a directory *inode* must be performed atomically. Each *inode* holds a semaphore which is locked and unlocked around these critical operations.

The HP-RT kernel uses the simple semaphore primitives *swait()* and *ssignal()* (corresponding to Dijkstra's P and V operations)[3] for process synchronization, mutual exclusion, and atomic resource management. A single 32-bit integer is used as a kernel semaphore data structure. This data structure supports two semaphore types: counting semaphores and priority-inheritance semaphores. With an additional level of lock and unlock code and using a separate integer as a counter, priority-inheritance semaphores can also be used as the basis for counting semaphores. Priority-inheritance semaphores are described later in this paper.

The semaphore primitives *ssignal* and *swait* have the code to interpret the contents of the kernel semaphore data structure and are able to differentiate between counting and priority-inheritance semaphores.

A counting semaphore in HP-RT holds a positive count value when the semaphore is unlocked and a resource is available. An *swait()* operation on a positive-valued semaphore causes the semaphore to be atomically decremented, and the calling thread continues execution. An *swait()* on a zero or negative-valued semaphore (the resource is not available) causes the thread to block (suspend) on the semaphore.

When one or more threads are blocked on a counting semaphore, the threads are placed into a priority-ordered linked list with the semaphore heading the list. To identify a semaphore that is locked and has one or more waiting threads, the semaphore is set to the negative address of the first waiting thread (see Fig. 6). The *sem* and *owner* fields shown in Fig. 6 are described below.

An *ssignal()* on an unlocked or locked-with-no-waiters counting semaphore merely causes the nonnegative value of the semaphore to be atomically incremented. An *ssignal()* on a locked semaphore with one or more waiters (one that holds a negative thread structure address) causes the first (highest-priority) waiting process to be unlinked and scheduled. Table 4 summarizes the different states of HP-RT counting semaphores.

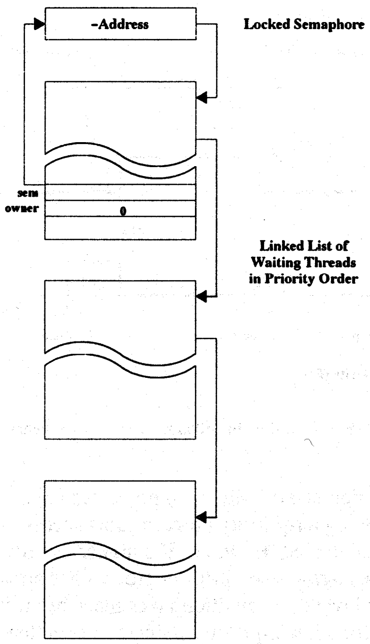


Fig. 6. A locked counting semaphore and waiting threads.

Table 4. Different States of Counting Semaphores

State	Meaning
0	Locked with no waiters
-Address	Locked with waiters (The address points to the first thread in the list of waiting threads.)
≥ 1	Unlocked

One drawback of this semaphore methodology is that there is no clear ownership of a locked semaphore. The second drawback is the risk of priority inversion.

Priority Inversion

In most real-time operating systems, a priority-driven preemptive scheduling approach is used. This scheduling method works well when a higher-priority process (or thread) can preempt a lower-priority process with no delays. One important problem that sometimes hampers the effectiveness of this scheduling algorithm is the problem of blocking caused by the synchronization of processes that share physical or logical resources.

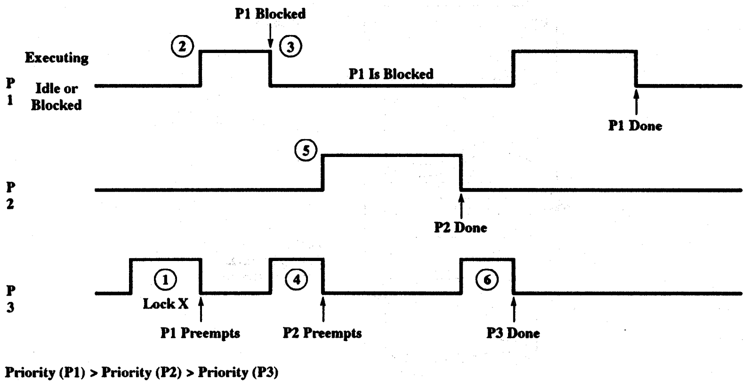


Fig. 7. A time line illustrating priority inversion.

The most common situation occurs when two processes attempt to access shared data. In a normal situation, if the higher-priority process gains access to the resource first, then good priority order is maintained. However, if a higher-priority process tries to gain access to a shared resource after a lower-priority process has already gained access to the resource, then a priority inversion condition takes place because the higher-priority process is required to wait for a lower-priority process to complete.

The following example, which is loosely based on an example first described by Lampson and Redell,[4] shows how a priority inversion can occur. Although the term process is used in the following example, the executing entity could just as well be a thread.

Let P1, P2, and P3 be three processes arranged in descending order of priority. Let processes P1 and P3 share a common data structure which is guarded by the binary semaphore X. Fig. 7 and the following sequence shows the events that can lead to a priority inversion:

1. P3 locks X and enters its critical section.
2. P1 arrives, preempts P3 and begins its processing.
3. P1 tries to lock X, but because X belongs to P3, P1 is blocked.
4. P3 again attempts to finish its critical section.
5. P2 arrives and preempts P3 before it finishes its critical section.
6. Assuming there are no more preemptions at some point P2 finishes, then P3 finishes, and P1 finally is unblocked on resource X and allowed to finish its critical section.

In this scenario the duration of P1's blocking is unpredictable because other processes can show up before P3 finishes its critical section and is able to release X.

Priority Inheritance

The methodology used in HP-RT to avoid the priority inversion problem employs priority-inheritance semaphores. The basic concept of priority-inheritance semaphores is that when process P blocks a higher-priority process, it executes its critical section at the highest priority level of all of the blocked jobs. Process P returns to its original priority level when it completes its critical section, which then allows the highest-priority blocked process to execute.

From the example above if P1 is blocked by P3 then according to the priority-inheritance concept, P3 inherits the same priority as P1 while it executes in its critical section. When process P2 arrives (while P3 is in its critical section) it would not be able to preempt process P3 because P3 would be running at a higher priority than P2. Thus, process P2 will not begin execution. When P3 finishes its critical section, process P1 can preempt P3 and run to completion. Then process P2 can begin execution.

Priority-inheritance semaphores can become quite complex when nested semaphore locks are allowed as they are in the HP-RT kernel. Not only must the current owner and all waiters for a semaphore be known, but given the owner of a particular semaphore, the highest-priority waiters of all semaphores currently owned by that owner must be known. This allows the system to manipulate priority properly as semaphores are released. The priority must revert to the priority of the current highest-priority waiter of all still-owned semaphores.

To manage this complexity and yet retain a single interface and data structure for semaphore operations, HP-RT uses the semaphore value –1 to indicate unlocked for a priority-inheritance semaphore. A value of one is not a possible thread structure address, so this value cannot be confused with the negative address of the first waiter of a counting semaphore.

Two fields in the thread structure are used to differentiate between the various states of priority-inheritance and counting semaphores when they are locked. A counting semaphore that is locked and has waiters will have the *sem* field in the first waiter thread holding the address of the semaphore and an *owner* field containing zero (see Fig. 6). A priority- inheritance semaphore that is locked and has no waiters will hold the negative address of the owner thread, which has a *sem* field with a value of zero (see Fig. 8a). Lastly, a locked priority-inheritance semaphore that has waiters will hold the negative address of the highest-priority waiting thread. This thread structure has a *sem* field holding the address of the semaphore and an owner field holding the address of the owning thread (see Fig. 8b).

To keep track of the highest-priority waiters for all owned priority-inheritance semaphores, a doubly linked list containing the highest-priority waiters for each owned semaphore is attached to the thread structure of each semaphore owner.

The different states of priority-inheritance semaphores are summarized in Table 5.

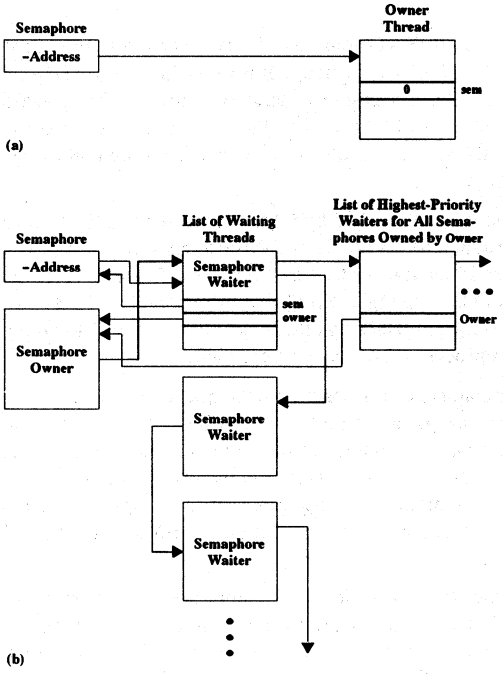

Semaphore
-Address
Owner Thread
0
sem
(a)
Semaphore
-Address
List of Waiting Threads
Semaphore Waiter
sem
owner
List of Highest-Priority Waiters for All Semaphores Owned by Owner
Owner
Semaphore Owner
Semaphore Waiter
Semaphore Waiter
(b)

Fig. 8. Data structures associated with priority-inheritance semaphores. (a) A locked semaphore with no waiting threads. (b) A locked semaphore with waiting threads.

Table 5. Different States of Priority-Inheritance Semaphores

State	Meaning
–1	Unlocked
–Address of thread owner	Locked without waiters (*sem* field in thread owner = 0)
–Address of highest-priority waiting thread (*sem* field in highest-priority waiting thread = semaphore address and *owner* field = thread owner address)	Locked with waiters

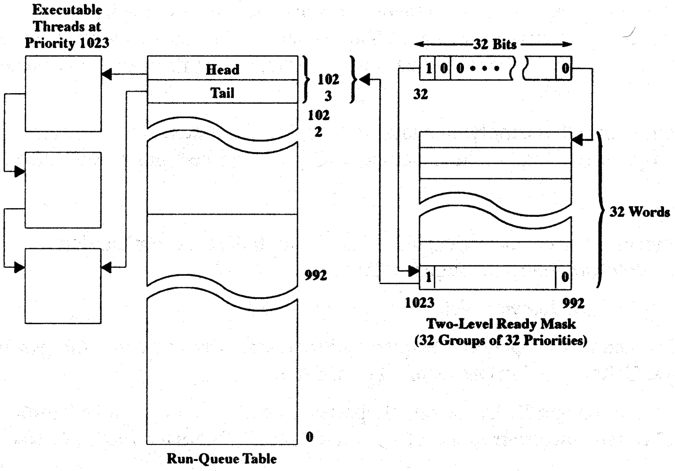


Fig. 9. Data structures for process scheduling in HP-RT.

Process Scheduling

HP-RT currently uses 64 distinct priority levels with the ability to extend support to 1024 distinct priority levels. Half of all HP-RT priorities are reserved for use by kernel management software. There is no explicit user program interface provided for placing a priority at these reserved levels. The reserved priorities are interleaved with the user priorities and are considered a "priority boost" on a user priority. Thus, between any two user priorities N and N + 1 lies a priority N + boost, which is more important than priority N and less important than priority N + 1. Boosted priorities are used by kernel service threads to provide service just above the priority of the highest-priority requesting process, but not at the next highest user priority which may be in use by the system user. Priority boosting is also used for temporary elevation of the priority of processes blocking on I/O operations to maximize throughput. This type of algorithm is only used in a user-specified portion of the overall priority range.

The HP-RT kernel internally manages priorities by converting from the user priority plus a possible boost value to a run queue table index by using the formula:

$$\text{Internal Priority} = (\text{user priority}) \times 2 + \text{boost},$$

where boost is either zero or one. Hence, if user priorities range from zero to 127, the internal priorities range from zero to 255.

HP-RT maintains a run-queue table with one entry per internal priority. Each entry holds a ready thread list head and a list tail pointer (see Fig. 9). To determine quickly the highest priority for which there is a runnable thread, HP-RT uses a two-level bit mask called a ready mask in which a set bit indicates a runnable thread. The top level of the ready mask is one 32-bit word. Each bit in this word indicates that within a set of 32 priorities, at least one thread is executable. Thus, if as shown in Fig. 9 the high-order bit of the first word of the ready mask is set, then there is at least one thread in the internal priority range of 1023 to 992 that is executable. The second level of the ready mask holds up to 32 32-bit entries each of which indicates which of these 32 priorities holds executable threads.

By using high-speed assembly language code to find the first set bit in the ready mask, the highest-priority thread in the nonempty run queue can be quickly determined.

References

1. B. Clements, "Mechanical Considerations for an Industrial Workstation," *Hewlett-Packard Journal,* August 1993.
2. *Hewlett-Packard Journal,* Vol. 41, no. 3, pp. 36-58.
3. E. W. Dijkstra, "Co-operating Sequential Processes," *Programming Languages,* F. Genuys, Editor, London: Academic Press, 1965.
4. B. W. Lampson and D. D. Redell, "Experiences with Processes and Monitors in Mesa," *Communications of the ACM,* Vol. 23, no. 2, February 1980, pp. 104-117.

HP-UX is based on and is compatible with UNIX System Laboratories' UNIX* operating system. It also complies with X/Open's* XPG3, POSIX 1003.1 and SVID2 interface specifications.

UNIX is a registered trademark of UNIX System Laboratories Inc. in the U.S.A. and other countries.

X/Open is a trademark of X/Open Company Limited in the UK and other countries.

Managing PA-RISC Machines for Real-Time Systems

Paper # 1010

by George A. Anzinger

Hewlett-Packard Company
11000 Wolfe Road MS-42UN
Cupertino, CA 95014-9804
(408)-447-5079

© 1993 Hewlett-Packard Company. Reprinted with permission.

In the HP-RT operating system, the interrupt-handling architecture is especially constructed to manage the high-performance timing requirements of real-time systems.

The task of an operating system is to manage the computer system's resources. This management should be done so as to give the best possible performance to user tasks or jobs presented to the system. How this performance is measured and valued differs depending on the task or mission of the system. The three major classes of tasks or missions presented to an operating system are timeshare, batch, and real time. The important aspects of performance of these three classifications differ, and, because they differ, require the operating system to use different algorithms to manage system resources.

Timeshare

Timeshare systems are usually designed to share system resources with all contending processes. The major resource to be shared is CPU time, which is usually sliced into small units (called time slices) and allocated to all runnable processes in a "fair" way. Various notions of fair exist and have been used, but in general, runnable processes contend at the same level or priority for CPU time. Some (or even most) systems modify this notion of fair to give more time to a process that blocks often and less to a process that is compute bound. Some systems may also have preferred priorities for processes that run on behalf of the system. Such processes may be handling printers, communication lines, or other things that are shared with several processes.

Batch

Batch systems are usually designed to maximize the throughput of the system. That is to say, they attempt to get the most work done in a given period of time. Such systems will not usually use a timeshare scheduling algorithm because it introduces overhead that does not add to the desired result—throughput. To help achieve maximum throughput, one popular batch scheduling algorithm is to run the job that has the least amount of time left to run. The point is that batch systems typically do not need to make any attempt to share CPU time.

Real Time

Real-time systems, unlike timeshare or batch systems, are usually designed to run the most important process that is ready. Importance is assigned by the user or designer of the system, and the operating system has little or nothing to say about it. The system designer (i.e., the user who sets up the system) decides the order of process importance and assigns priorities for all processes on the system. The operating system's job then is very simple: give the CPU to the highest-priority process that is ready. The performance of a real-time system is usually measured by how fast it can respond to events that change the identity of the highest-priority ready process. Such events are usually external and come to the system in the form of interrupts, but can also be internal in the form of processes that promote other processes to higher priorities (or demote themselves to lower priorities). Another major event that real-time systems must respond to is the passage of time. The indication of the passage of time also comes to the system in the form of an external interrupt.

From this discussion, it is apparent that one major measure of a real-time system is how quickly it can respond to an interrupt. A response consists of:

- Recognizing that the interrupt is pending
- Processing the interrupt (i.e., deciding what to do about it)
- Taking the indicated action.

Usually the indicated action will be to switch context to the process that is to handle the interrupt. Context switching encompasses the actions taken when control or execution moves from one process to another as a result of an interrupt or some other event (see "Context Switching in HP-RT" on page 1010-3 for more about context switching).

From a system's point of view the response (or response time) is the time it takes the whole system† to do something that changes the environment it is monitoring or controlling. From an operating system vendor's point of view the response stops when the user code gets control and the operating system's responsiveness is no longer key to system performance.

While the system is dealing with one interrupt and preparing a response, it may need to contend with other interrupts that are less urgent. The system must take the time to determine this.

It is also possible that, at the time an interrupt arrives, the system is in a state in which the interrupt system or context switching is off. The system needs to go into these states to protect shared data from corruption by contending processes (see "Protecting Shared Data Structures," on page 1010-6). Some systems protect themselves and their shared data by turning off context switching whenever they are in system code.

This is not reasonable for a high-performance real-time system that is trying to switch contexts in less than 50 μs. For these systems it is necessary to recognize and process interrupts in the 25-μs range. This implies that the interrupt off time plus the interrupt processing time must be kept below 25 μs.

This paper will explore the problems a PA-RISC architecture presents to real-time processing. These problems revolve around the need for fast context switching, interrupt handling, and repeatability. Next, possible solutions to these problems will be discussed, detailing the solutions used in the HP-RT (real-time) operating system, which runs on the HP 9000 Model 742rt VMEbus board computer. The hardware and software components of the Model 742rt are described in paper 1008.

†This includes the operating system, the user application, and the external devices.

Context Switching in HP-RT

Context switching can be defined as moving abruptly from one area of code to another as the direct result of some influence outside of the program or programs being switched to or from. Usually the context switch is the direct result of an interrupt or trap (a trap is an internal interrupt caused by some program activity such as divide by zero or illegal memory access). A context switch can also occur as a result of a program or thread blocking. In this case the operating system will context switch to a program or thread that is not blocked. These two different ways of generating a context switch have different overhead costs as will be explained below. One of the measures of a real-time system is how fast it context switches. When used in this way the reference is to how fast one user process can be suspended and another user process restarted.

To context switch, the operating system must save the *from* process's state. The state consists of all the machine registers that the program may depend on. After saving the *from* process state, the *to* process's state must be restored. As a result of this save and restore, both the *to* and *from* processes have their view of the world preserved and restored respectively even if they are suspended for a very long time.

For example, consider the case of a user program that has asked for some device input. The program will be suspended or blocked on the device driver waiting for the device to respond with the desired data. While waiting, the operating system will find some other program that is ready to run and switch to it. When the desired data arrives, the processor will be interrupted and the operating system will switch control of the processor to the waiting program.

As an example of a context switch that is strictly the result of an external interrupt, consider the case in which a time slice is exhausted. In this case, both the program being switched from and the one being switched to are interrupted as opposed to having to block and wait for a resource.

From a system overhead point of view there are four different types of context switch:

- Both the *from* and the *to* processes enter the blocked state programmatically
- The *from* process blocks programmatically and the *to* process is interrupted
- The *from* process is interrupted and the *to* process is blocked programmatically
- Both processes are interrupted.

Because of calling sequence conventions, processes that are interrupted incur additional overhead to save and restore caller registers.

To take advantage of the savings possible when processes block programmatically, HP-RT uses a context switch routine based on this type of block. The extra work required when processes are interrupted is performed by code in the system interrupt handler.

PA-RISC Architecture

The RISC architecture is used to speed up CPUs by designing them so that each instruction is simple and can be executed quickly. The goal is usually to have each instruction take the same amount of time to execute and to design the machine so that several instructions can be pipelined. To get all instructions to execute in the same time requires that no one instruction can be complex. Operations that are complex and require more than one instruction time are either handled by subroutines or by coprocessors. Coprocessors are designed to run independently allowing the main processor to do other useful work while the coprocessor does its work. For example, HP's PA-RISC machines use coprocessors to do floating-point math.

In HP's PA-RISC processors, the following characteristics are important for real-time applications:

- Memory reference instructions either load or store and do nothing else. This means that there is no read-modify-write instruction.
- Memory reference instructions may stall if the data is not available. To help in this regard, a cache memory is used to speed up the average access to memory.
- Since memory accesses are potential roadblocks, 32 general-purpose registers are available as well as 27 control registers and 32 64-bit-wide floating-point registers. This allows the processor to keep most of the variables of interest in registers, avoiding slow memory access operations.
- All interrupt context is kept in control registers.

Real Time and HP's PA-RISC

From a real-time perspective, the characteristics of HP's PA-RISC that are of concern are those that limit performance in the real-time sense. As discussed above, a real-time system must be able to change its mind (context switch) quickly. This implies that the large context associated with a process can be a problem. Also, while changing context, as well as doing other things, the system needs to be even more responsive to interrupts. This means we must not turn the interrupt system off for long times. In particular, we must not turn it off for the duration of a context switch.

HP-RT is the result of porting a third-party operating system† to the HP 9000 Model 742rt board level real-time computer.

As such, the porting team was constrained to work with the conventions existing in the system being ported. Likewise, the porting team was not empowered to change any of the language or hardware conventions that exist in HP's PA-RISC machines and the HP-UX* host operating system.

To take advantage of the best of HP's PA-RISC processors, the port team decided to restrict the system to PA-RISC 1.1 architectures. The 1.1 architecture provides shadow registers that allow system interrupt code to be run without saving any context (see "The Shadow Register Environment," on page 1010-8).

On examining the way the system we were porting recommends that drivers be written we found the following:

†LynxOS from Lynx Real-Time Systems Inc.

- After an interrupt, the system enters the interrupt service routine. The routine should be written in C and should make a call to the operating system function *ssignal* and then return.
- The function *ssignal* increments a counting semaphore, and if the result is 0, the interrupt service thread is put in the ready list (execution threads and counting semaphores used in the HP-RT operating system are described in paper 1008).
- If the new entry in the ready list has a higher priority than the current process, a flag is set indicating that a context switch is needed. (Context cannot be switched while in an interrupt handler.)
- When the driver's interrupt service routine returns, the system notices whether a context switch is pending and if so takes the required action. If not, the system just returns to the point of the interrupt.

The problem with this picture is that to call the interrupt service routine the system has to save most of the system state. This is a lot of overhead for only one function call and return.

The team decided that a better way to handle interrupt servicing would be to code a companion *ssignal* function. The new *ssignal* runs using only the shadow registers and still does everything the original *ssignal* did. This scheme allows the whole *ssignal* call to be made without establishing a C context, which involves saving and restoring the C environment (see "C Environment," on page 1010-11). However, some restrictions are placed on I/O drivers in that they have to make their semaphores known to the operating system.

In some cases, calling the *ssignal* function is almost all that an interrupt service routine will do. It is also possible that a few lines of assembly code might be required to complete the interrupt service routine. Such code might move a byte of incoming data from the I/O device to an internal buffer. For applications that have these kinds of interrupts, the system provides the ability to call an assembly language interrupt service routine. To keep overhead low, the assembly language interrupt routine is restricted to using the shadow registers and no system resources. The system interrupt dispatcher calls the *ssignal* function if the assembly language routine returns a nonzero semaphore.

Some I/O devices and drivers require full C-code interrupt handlers. For these interrupts, the system establishes a C context on an interrupt control stack. In this context interrupts of higher priority are turned on while the interrupt is processed. These routines can also call a limited number of system functions. For example, the system time base interrupt is handled by a C interrupt handler.

With three different possible interrupt handling situations, the operating system needs to have the ability to decide quickly which interrupt service routine to use. Usually this is done by either a table index, in which the system determines the method to use via a number that is an index into a table of routines to call, or a case statement, in which the indicated method, again expressed as a number, is used to indicate which code to execute. A much quicker method than these two is to put the address of the interrupt service routine in the driver's table structure. This also allows the system to be expanded easily to handle other interrupt handler environments.

Protecting Shared Data Structures

Shared data structures are needed in any operating system to keep track of the resources that the system is sharing among several processes. For example, each process will need memory for its code and data. This memory is a shared resource and the management structures must be accessed in a way that will not allow the system to lose parts of the resource. One method of keeping track of a resource like memory is to keep free pages of memory in a free list. When a page of memory is needed, the page at the head of the free list is removed from the list and given to the requesting process. This removal (and its subsequent return) must be done in an atomic operation with respect to the contending processes. By this we mean that, as far as any process cares, the removal of a page from the free list happens as one indivisible operation. Otherwise, a contending process could get control and possibly get the same page.

The importance of maintaining atomicity in dealing with a shared resource such as memory on a free list is illustrated in the following example. The process of removing page A from the free list involves:

1. Picking up the pointer to page A from the list head
2. Using the resulting pointer to get the pointer to page B, which is in the first word of page A
3. Storing the pointer to page B in the list head.

If the removal is interrupted after step 1 but before step 3, and the interrupting process also tries to remove a page from the free list, both processes will get the same page and most likely the system will fail. Similar problems on returning of pages to the free list can result in lost pages or even circular free lists.

The solution to these problems is to make a sensitive operation atomic with respect to contenders. If only processes can contend, it is sufficient to prevent context switches for these periods of time. If one or more of the contenders runs on an interrupt, then interrupts must be disabled to achieve the required atomic operation.

The HP-RT system supports three levels of contention protection:

- Interrupts disabled
- Context switch disabled
- Semaphore locking.

From an overhead point of view, the cost is lowest for the interrupt disable and highest for the semaphore lock. From an impact on performance point of view, interrupts should be disabled only for short periods of time, context switch disabled only for slightly longer times, and semaphores held as long as needed.

For short operations, such as the list removal operation described above, the interrupt disable method is the best to use (even if the atomic test does not require this level of protection) because the disable time is short and the overhead of interrupt disable protection is the lowest of the three methods.

A New Interrupt Environment

The need to deal with the three interrupt handling situations described above and the requirement to handle interrupts from the VMEbus meant that we had to design and implement a new interrupt handling environment. Fig. 1 shows a simplified view of the logical I/O architecture that the HP-RT interrupt handling subsystem is designed to service.

The nature of the VMEbus requires a second level of interrupt dispatch. This is necessary because VMEbus interrupts come into the PA-RISC processor via one of seven lines or PA-RISC interrupt levels. As shown in Fig. 1, each of these lines can handle several independent devices, which implies several interrupts.

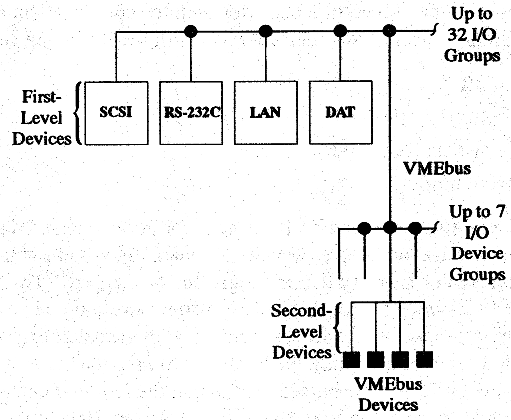


Fig. 1. A logical view of the I/O architecture the HP-RT operating system is designed to work with.

The VMEbus standard specifies that a device requesting an interrupt must assert its request on the interrupt line it is using. The interrupt responder sees the request and sends back an interrupt acknowledgment for that interrupt line. Each device using the same line blocks the acknowledgment signal from being seen by devices farther away from slot 0† while it has an interrupt request pending. When a device with an interrupt pending sees an interrupt acknowledge it responds by sending back an interrupt vector. The interrupt vector is a data element (byte or word) that identifies the interrupting device and is used by the interrupt responder to dispatch the interrupt.

†Slot 0 in a VMEbus cardcage typically houses the card or cards that contain the VMEbus system controller and other resources.

The Shadow Register Environment

The PA-RISC 1.1 implementation added shadow registers to the basic machine architecture. Shadow registers are seven registers into which the contents of GRs (general registers) 1, 8, 9, 16, 17, 24, and 25 are copied upon interruption. The contents of these general registers are restored from their shadow registers when an *RFIR* (return from interruption and restore) instruction is executed.

The shadow register environment includes code that executes between a processor interrupt and the following *RFIR* instruction. This code is executed in HP-RT using only the shadow registers. It is important to note that the nature of this environment is further defined by the nature of the processor's behavior on interrupt. When an interrupt occurs the processor transfers control to the interrupt code with the following state:

- Interrupt system off
- Interrupt state collection disabled
- Virtual memory system (both code and data) off
- All access protection off.

Since the virtual memory system is off, all memory for both code and data must reside in and be accessed by physical addresses. Usually an operating system will put the interrupt handling code in an area of memory that is "equivalently mapped." This means that the physical and virtual addresses are the same. This also means that code running in the shadow register environment cannot access memory with virtual addresses that are not equivalent since to do so would require the hardware to map the address using its TLB (translation lookaside buffer).† The hazard here is that the required entry may not be in the TLB, which would cause a trap to the TLB miss handler. Since traps are a form of interrupt, the miss handler would not be provided with the interrupt state (because the interrupt state collection is disabled) and thus would not know how to return to the point of the trap.

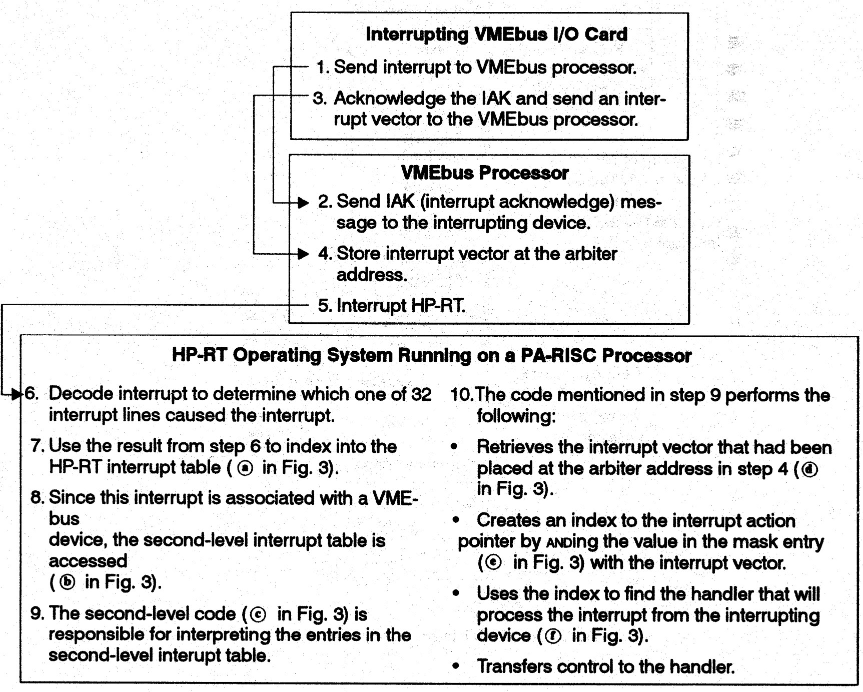
On the plus side, if the whole interrupt can be processed in the shadow register environment, the *RFIR* instruction is all that is needed to return to the point of interruption.

---

†The translation lookaside buffer or TLB is a hardware address translation table. The TLB speeds up virtual-to-real address translations by acting as a cache for recent translations.

The original plan for the Model 742rt hardware was to interrupt the PA-RISC processor when a VMEbus interrupt request was asserted and to do the interrupt acknowledgment when the processor attempted to read the interrupt vector. This plan required the operating system to stall in the interrupt handler with the interrupt system off for an unspecified length of time because VMEbus devices are not required to yield the bus to a requester, making the actual time required to do an operation on the bus open-ended. To solve this problem, the HP-RT team decided that the interrupt vector should be prefetched by the hardware before interrupting the PA-RISC processor. In this way a VMEbus interrupt can be dispatched without the PA-RISC processor having to wait for the VMEbus processor to fetch the interrupt vector. The current hardware always does the interrupt acknowledge as soon as possible but has the option of asserting the processor interrupt either immediately or on completion of the interrupt acknowledgment.

Fig. 2 shows the steps involved in handling a VMEbus interrupt and Fig. 3 shows a portion of the system interrupt table which is used for handling second-level VMEbus interrupts and non-VMEbus interrupts. Note the correspondence between the interrupt table structure and logical I/O architecture shown in Fig. 1. The three different interrupt handling situations mentioned above are taken care of by allowing one of the three types of interrupt routines to be specified in the table (see the interrupt action entry in Fig. 3).



Interrupting VMEbus I/O Card

1. Send interrupt to VMEbus processor.
3. Acknowledge the IAK and send an interrupt vector to the VMEbus processor.

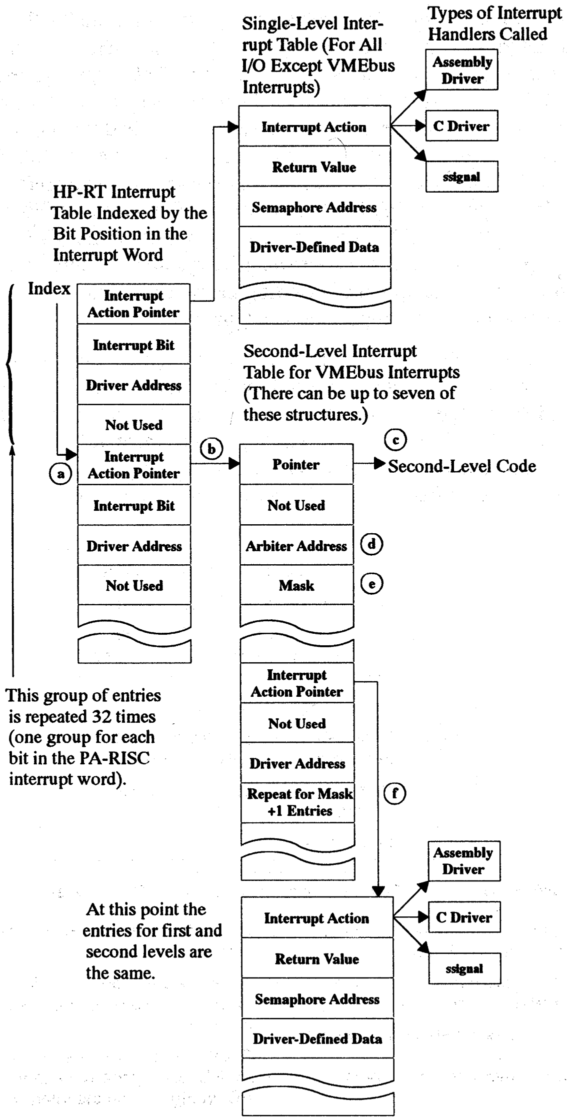
VMEbus Processor

2. Send IAK (interrupt acknowledge) message to the interrupting device.
4. Store interrupt vector at the arbiter address.
5. Interrupt HP-RT.

HP-RT Operating System Running on a PA-RISC Processor

6. Decode interrupt to determine which one of 32 interrupt lines caused the interrupt.
7. Use the result from step 6 to index into the HP-RT interrupt table (ⓐ in Fig. 3).
8. Since this interrupt is associated with a VMEbus device, the second-level interrupt table is accessed (ⓑ in Fig. 3).
9. The second-level code (ⓒ in Fig. 3) is responsible for interpreting the entries in the second-level interupt table.
10. The code mentioned in step 9 performs the following:
    - Retrieves the interrupt vector that had been placed at the arbiter address in step 4 (ⓓ in Fig. 3).
    - Creates an index to the interrupt action pointer by ANDing the value in the mask entry (ⓔ in Fig. 3) with the interrupt vector.
    - Uses the index to find the handler that will process the interrupt from the interrupting device (ⓕ in Fig. 3).
    - Transfers control to the handler.

Fig. 2. An example of the VMEbus interrupt handling process.


Types of Interrupt Handlers Called
Assembly Driver
C Driver
ssignal
Interrupt Action
Return Value
Semaphore Address
Driver-Defined Data
Index
Interrupt Action Pointer
Interrupt Bit
Driver Address
Not Used
a
b
c
d
e
f
Pointer
Not Used
Arbiter Address
Mask
Second-Level Code
Interrupt Action Pointer
Not Used
Driver Address
Repeat for Mask +1 Entries
Assembly Driver
C Driver
ssignal
Interrupt Action
Return Value
Semaphore Address
Driver-Defined Data

Single-Level Interrupt Table (For All I/O Except VMEbus Interrupts)

HP-RT Interrupt Table Indexed by the Bit Position in the Interrupt Word

Second-Level Interrupt Table for VMEbus Interrupts (There can be up to seven of these structures.)

This group of entries is repeated 32 times (one group for each bit in the PA-RISC interrupt word).

At this point the entries for first and second levels are the same.

Fig. 3. The HP-RT interrupt table structure.

Second-level VMEbus interrupts are handled by reading the returned interrupt vector, masking it, and using the result to index to the interrupt action that will handle the interrupt (ⓕ in Fig. 3). The masking is done to prevent indexing to a location outside of the table and to allow the interrupting device to pass back status information in the high part of the word. The mask is computed at system configuration time from the user's specification of the high number to be returned on a given interrupt line. This number is rounded up to the nearest power of two (2^n). For example, if the highest number to be returned on a particular interrupt line is 12 then n is four because 2^4 provides the nearest power of two greater than 12.† This results in a table that is larger than needed but eliminates the need to check if the masked number is too large. Unused entries in both the first-level and second-level interrupt tables are filled with entries that result in system illegal interrupt messages should such an interrupt ever happen.

Initially, the HP-RT team wanted the interrupt handler and the interrupt off times to be "blind" to interrupts for a maximum of 100 instruction times, including any stall states minus cache misses. The notion of blind to interrupts was introduced to cover the case in which the system keeps the interrupt system off, but still processes the interrupt in a timely fashion. This occurs in the interrupt handler, for example, when after it processes an interrupt it looks at the pending interrupts and if it finds one, processes it without turning on the interrupt system. The operating system interrupt dispatching code met the 100-instruction time limit.

C Environment

C environment refers to the implied machine state when executing in a C language program. This machine state is really a set of register use conventions that are defined in the software architecture for the PA-RISC processors (see Fig. 4). Some of the basic assumptions made in C about these registers include:

- Register 30 is the stack pointer and points at the first available double word on the stack. The stack grows with increasing addresses.
- Just below the current stack pointer is a standard stack frame with room for the return address to be saved (if the callee needs to save it) and room for each of the call parameters to be saved.
- Registers 26, 25, 24, and 23 (as needed) contain the call arguments. If more than four arguments are passed, those above the first four arguments are stored in the stack frame.
- Register 27 is the global data register and is used to address any global data needed by the procedure.
- Register 2 contains the address to return to when the procedure is done.
- Registers 28 and if needed 29 are to contain the function result when the function returns.
- Registers 3 through 18 (the callee-saves registers) can be used only if they are saved and restored before returning to the caller.
- Registers 19 through 22 (the caller-saves registers) and registers 1 and 31 are available to use as scratch registers.

There are other conventions for floating-point and space registers which are usually not important in operating system code.

The shadow register environment, which consists of registers 1, 8, 9, 16, 17, 24, and 25, is not compatible with the C environment.

GR0	**Zero (by Hardware Convention)**
GR1	**Scratch**
GR2	**RP (Return Pointer)**
GR3 ⋮ GR18	**Callee-Saves Registers**
GR19 ⋮ GR22	**Caller-Saves Registers**
GR23 ⋮ GR26	**Arguments**
GR27	**DP (Global Data Pointer)**
GR28 GR29	**Return Values**
GR30	**SP (Stack Pointer)**
GR31	**MRP (Millicode Return Pointer)**

Fig. 4. Register use conventions in the C environment.

Handling Large Contexts

The PA-RISC architecture divides a program's context into two register sets: *caller-saves* and *callee-saves* registers. The caller-saves registers consist of registers that are expected to contain values that do not need to be preserved across a procedure call, that is, values the calling function does not care about. Therefore, these registers are available for use as scratch registers or for parameter passing by the called routine. The callee-saves registers are used for values that must be preserved across a procedure call. Thus, if the called routine wants to use a callee-saves register, it must first save it and then restore it before it returns. The PA-RISC architecture also specifies where these registers must be saved on the call stack (see Fig. 5). This caller-saves and callee-saves convention is used by the PA-RISC compilers so that the system can depend on it.

HP-RT depends on the caller-saves and callee-saves division to keep context management code to a minimum. In particular, on system calls the system saves only the user's (caller's) return address, global register, and stack pointer. The system call handler then calls the requested system call function depending on that function to save and restore any callee-saves registers it may want to use. Likewise, on interrupts or traps where control must be transferred to the kernel stack, only the caller-saves registers need to be saved because HP-RT depends on callee-saves registers to be saved by any function called. Therefore, since the context switch code is called as a function, all it has to save are the callee-saves registers. By saving only what needs to be saved at each step, the system keeps the overhead low for register saves and restores.

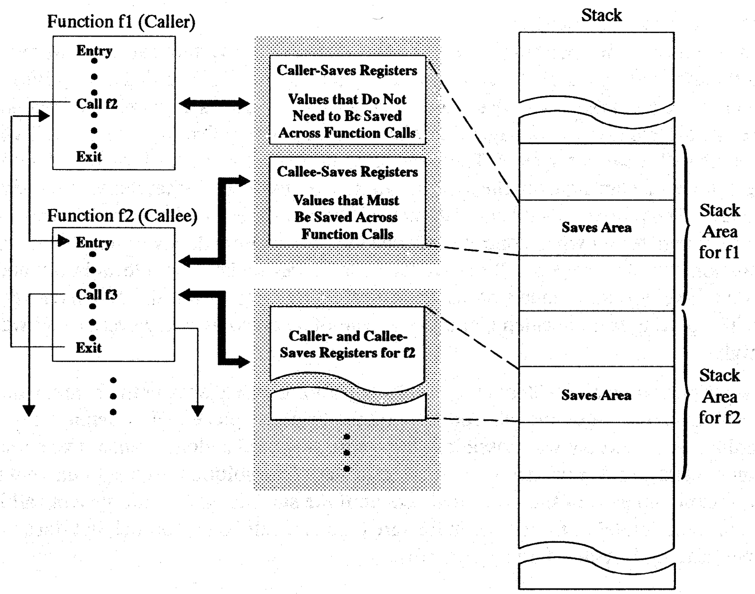

Function f1 (Caller)
Entry
Call f2
Exit
Function f2 (Callee)
Entry
Call f3
Exit
Caller-Saves Registers
Values that Do Not Need to Be Saved Across Function Calls
Callee-Saves Registers
Values that Must Be Saved Across Function Calls
Caller- and Callee-Saves Registers for f2
Stack
Saves Area
Stack Area for f1
Saves Area
Stack Area for f2

Fig. 5. The relationships between function (or procedure) calls, the caller- and callee-saves registers, and the stack area. The caller puts data it wants to preserve in the callee-saves registers before making a call. If the called routine (callee) needs to use any of the callee-saves registers, it saves the value contained in the register and restores the value back into the register before returning to the caller.

HP-RT also takes advantage of the fact that the floating-point coprocessor is enabled by setting bits in a control register. If the coprocessor is not enabled, the system will generate an emulation trap when a process attempts to use any floating-point instructions. Processes start with the floating-point coprocessor disabled. When a process attempts to use floating-point instructions, the code in the emulation trap handler saves the old process's floating-point registers and loads the current process's floating-point registers. In this way, the overhead of floating-point context switching is limited to only the times when it is needed.

In deference to maintaining a low interrupt-off time, the system checks for pending interrupts once it has stored the old process's floating-point registers. If any external interrupts are pending at this time, the system will set the floating-point ownership flags to show that the coprocessor is not owned and then handle the interrupt. The current process will be redispatched still not owning the floating-point coprocessor, but will immediately end up in the emulation trap which will finish the context switch. Of course the interrupt could cause the current process to lose the CPU, possibly even to the process whose state the system just saved. For this reason, a flag is kept to show that the registers were not changed so the process may proceed with only a quick pass through the emulation code to get the coprocessor bits set again.

Setjmp and *Longjmp* Solutions

On rare occasions the operating system is required to abort a system call. This occurs when the user sets up a signal handler and the signal handler is specified as requiring the termination of any system call that is pending when the signal is delivered. As mentioned above, the system takes advantage of the fact that functions called on a system call will restore the callee-saves registers. These registers are saved on the stack by each function in the call chain, starting from the system call handler to the code that delivers the signal to the user. The problem then is how to recover these registers so the user code will have the correct register set when control is returned to it. The normal way to handle this kind of situation is to do a *setjmp* call to save the callee-saves registers in a local buffer and then do a *longjmp* call (which restores the saved registers) from the signal delivering code. The porting team decided that the overhead of a *setjmp* on every system call was too high.

One solution that was considered was to identify all possible places in the kernel where such a signal could be delivered. Code could then be put in place to do a *setjmp* only when the signal delivery was possible. This approach was abandoned when it was found that these calls could come from user-written drivers. The solution used is to unwind the stack, picking up each of the saved registers until the stack is back to the system call handler. This solution takes more time in the rare case of a call being aborted, but does not put overhead in the path of all system calls.

Hardware Help

It was mentioned above that the VMEbus hardware holds off interrupts until the information needed to process the interrupt is available. The HP-RT team also requested and received a real-time mode in the interrupt convention for onboard I/O device interrupts. The normal convention was that all onboard device interrupts were collected into one bit (each bit corresponds to one interrupt line). Under this convention the software interrupt handler would first decode the interrupt source to this bit and then read an I/O space register that contained a bit map of all the onboard devices requesting interrupt service. The hardware convention used was to clear this register when it was read. This required the software to keep track of all the bits that were set and to call the handler for each bit. The software management task for this convention would have been fairly high because the real-time system wants the interrupt system on most of the time, which means that it is possible for another interrupt to be received from another onboard device before the current interrupt is completely processed. At the same time, the rest of the main processor's interrupt register would not be in use.

The HP-RT team asked for an interrupt mode in which each onboard device has its own interrupt bit on which it can interrupt the main processor. This convention not only eliminates the need to remember which bits were set, but also eliminates a level of decoding in the interrupt path.

Conclusion

One of the main goals of the HP-RT project was to minimize the time to handle interrupts. Table I, which shows the results of these efforts, is a task response time line that shows the time consumed by each activity in the path from an interrupt to the task (e.g., user code) that does something to respond to the interrupt. For cases in which an interrupt is handled by an interrupt service routine in the operating system and not user code, the interrupts disabled and dispatch interrupts times shown in Table I are the only times involved in determining the total task response time. Their worst-case times in this situation are 80 μs and 6 μs respectively, giving a total task response time of 86 μs. The 80 μs time is rare and work is continuing to reduce this time.

Table 1. Time Line for HP-RT Running on the HP 9000 Model 742rt

Tasks Performed After an External Event	Task Response Best Case	 Worst Case
Interrupts disabled	0	0
Dispatch interrupts	3 μs	6 μs
Other interrupts	0	9 μs†
Context switch off	0	166 μs††
Scheduling and switching	27 μs	45 μs
Return from system call	1.2 μs	4.6 μs
Total time	31.2 μs	230.6 μs

† Three interrupts

†† This time is rare and is in code other than the interrupt and context switch code. Work is continuing to reduce this time.

HP-UX is based on and is compatible with UNIX System Laboratories' UNIX* operating system. It also complies with X/Open's* XPG3, POSIX 1003.1 and SVID2 interface specifications.

UNIX is a registered trademark of UNIX System Laboratories Inc. in the U.S.A. and other countries.

X/Open is a trademark of X/Open Company Limited in the UK and other countries.

Paper # 1011

RTE to UNIX Migration Tools & Techniques

Bob Combs
Combs International, Inc.
886 Belmont Avenue, Suite 3
North Haledon, NJ 07508

(201) 427-9292

Abstract

Most programmers who have moved applications from the 16-bit RTE environment to the 32-bit HP-UX (or other UNIX) environment have experienced significant surprises hidden in their applications, or at least in the differences in the operating systems. This paper will describe currently existing tools and techniques to minimize both the surprises and the time migration actually takes.

It will review the migration tools already in the HP-UX CSL, including RTE file routines, the RTE library, and the Migration Analysis Utility (MAU).

Discussion of some typical migration situations will show the size of typical conversion projects. Strategies will include bringing up applications fast. Some discussion will reflect on moving to a graphical user interface (GUI) and its implications.

1. Overview

In the 1980's Hewlett Packard introduced the 9000 800 Series computer. It sported a new operating system called HP-UX; HP's flavor of UNIX. RTE Users were told that this was the replacement machine for the HP1000's, and that the HP-UX machines would support real-time applications. Keep in mind that the folks making these claims did not come from the real-time portion of HP, and probably thought real-time meant the opposite of batch job submission.

Consequently, an RTE look-alike environment was offered by HP called PORT/RX. In many respects it did a fair rendition of an RTE. But because it was mostly an emulator with shadow files and other run-time atrocities, users found it actually ran slower than the slowest HP1000. Worse yet, real-time applications couldn't run with any deterministic assurances. HP's claims of real-time functionality in HP-UX became a nasty joke to the RTE Users. And to this day, most real-time aficionados consider UNIX just that; a nasty joke.

In fairness, there was a very useful side to poor PORT/RX. It had a Migration Analysis Utility (MAU), and a native mode library of many of the common RTE Library routines. HP, mostly due to the bad press, decided to deep six the PORT/RX product. But fortunately some saw its usefulness and got HP to contribute the workable portions to the INTEREX HP-UX CSL.

Some years have passed since the introduction of HP-UX, and most RTE Users now accept UNIX; either of their own free will, or by their management's "free will". True PA-RISC machines have gotten faster, but HP-UX still isn't real-time. While for many applications it may not matter, for some it is a critical issue holding them to the 16-bit architecture of the HP1000. Enter HP-RT.

HP-RT is a real-time UNIX which has come from the same lab folks in HP who brought us RTE. It operates on PA-RISC architecture and boasts real-time response beyond that achievable by RTE systems.

So, there is life after RTE, and now we have choices.

The biggest software headache is still "How do I migrate my code to UNIX with the least amount of effort?". Its this least amount of effort which we'll focus on below.

2. Planning the Migration

Planning a migration starts with a fundamental question about the need for real-time. This will define whether to migrate to HP-UX or HP-RT. If *hard* real-time is required (response times below tenths of a second), then you will need to migrate to HP-RT. Otherwise HP-UX will probably suffice.

Most RTE users have written their applications in FORTRAN. We'll be focusing on how to migrate FORTRAN, but C and PASCAL can also be migrated with the same level of ease. Obviously any code written in MACRO will have to be rewritten from scratch.

Since many have known they were eventually going to migrate, they have been writing their applications in such a way as to localize the RTE connections to a few central routines. This reduces the amount of migration recoding needed. Centralized modules are generally used for input/output, program scheduling, and program-to-program communication.

2.1. Step 1

Determine the statistics of your package. How many programs? How many modules? How many lines of code?

Use MAU to determine the changes and how many will need to be made. With these numbers, you can estimate how much time is required to modify code.

Before you make your estimates, scan through the MAU output, ignoring any flagged code that you can use library routines from the libraries listed below. The estimate should be made on the MAU numbers less the count of what can be recovered from the libraries.

2.2. Step 2

Set your code conversion standards. The first part of this is to set a list of the language syntax that needs to be changed. For example, use UNIX's sed (streams editor) to delete, add, or modify certain FORTRAN statements which are different from RTE to UNIX.

The second part is to examine the MAU outputs and begin developing routines or code sequences that can be used to replace those calls which cannot be obtained from the migration libraries. This should include program scheduling calls, program-to-program communication, etc.

2.3. Step 3

Determine the implementation sequence. Your specific application will dictate what has to be implemented in what order.

For example, if the application is mostly screen I/O, you'll need to get your screen I/O routines debugged first to be able to debug the application processing.

2.4. Step 4

Do it.

3. RTE versus UNIX

There are many glaring differences between RTE and UNIX. For example, let's look at the scheduling of programs.

3.1. Real-time Scheduling

RTE allows programs to schedule in a time-sliced manner for background, or as a foreground high priority. We'll define foreground as the high priority (low number) area where a process is below both the time slice fence and the background fence. Real-time programs use the foreground since it allows them to schedule based upon some event. When that event occurs, all other programs are pre-emptied until the real-time program has completed it processing.

HP-UX has a background and "real-time" processing capability, but while its background scheduling operates similar to RTE's, its real-time scheduling is not able to truly pre-empt upon an event. The HP-UX system still waits until the next time-slice clock interrupt to check if a "real-time" process is requesting control. Another flaw in HP-UX's real-time capabilities is that multiple "real-time" processes at the same priority are time sliced. The duration of the time slice period for each "real-time" priority is configurable.

3.2. Address Space

Since RTE is a 16-bit machine and UNIX is a 32-bit machine, it comes as no surprise that overlays are not needed when moving programs to UNIX.

Also, sharing memory under RTE requires either system COMMON or Shared EMA. UNIX uses shared memory which is part of the address space of a program. Therefore, UNIX's shared memory is quicker to access than RTE's SHEMA, but coding changes are required.

Most of programs use 16-bit integers (Integer*2). The easiest migration will be to not attempt to convert to 32-bit but to retain the 16-bit structure.

3.3. EXEC Calls

The most obvious difference is the EXEC calls, which under RTE are used for most system level control or low-level I/O requests. The EXEC call under HP-UX will call for another program to be loaded over your current program. The two types of calls are not compatible.

Here are some suggestions to examine as replacement routines for the RTE EXEC calls.

RTE function	RTE EXEC #	HP-UX routine suggestion
read	EXEC 1	fread or fgets
write	EXEC 2	fwrite or puts
i/o control	EXEC 3	ioctl
terminate	EXEC 6	exit
schedule	EXEC 9	fork & exec & wait
schedule w/o wait	EXEC 10	fork & exec
get time	EXEC 11	time
read/write msg	EXEC 20	msgsnd, msgrcv
setup msg queue	EXEC 21	msgget, msgctl

3.4. Character Packing

The HP1000 packs characters into the left (upper) byte first and then the right (lower) byte. Most other machines, including PA-RISC pack from right to left. Fortunately, the compilers will correct most of this by simply recompiling on the HP-UX machine. It is something that should be watched for if programs are masking certain bytes. For example,

```
        C=IAND(ISHFT(WD,-8),O'377')
```

will generate a bug since the first character is in the right byte, not the left.

4. FORTRAN Code

There are several differences in FORTRAN under RTE versus FORTRAN under HP-UX.

4.1. Control Line

The control line is not used under UX. Delete the line "FTN77" etc.

4.2. Options

Since UX is already code and data separated, the option "$CDS ON" is not needed, and will not be recognized by the UX compiler.

In fact, most of the $ options are different.

There are two options which should be added to the top of the source modules,

```
        $SHORT
        $HP1000 ARRAYS
```

The "$SHORT" option tells the compiler that the default value size will be 16-bit (just like RTE). The $HP1000 ARRAYS" option tells the compiler to handle arrays as they are under RTE.

By default, the UX compiler will accept up to 19 continuations of a line. Programs that have more than 19 continuation lines will need to add the option to set the number of lines higher. For example,

```
        $CONTINUATIONS 99
```

4.3. EMA

The RTE compiler option for EMA,

```
        $EMA /label/
```

can be converted to UX shared memory by changing to the UX option

```
        $SHARED_COMMON KEY="lbl1" /label/
```

Note that the key value "lbl1" is the actual shared memory key.

4.4. PROGRAM

RTE can have a program type and priority, plus a relocatable comment field with the program statement, i.e.

```
        PROGRAM MYPRG(3,99), my prog <930624.1426>
```

Since UX does not support these, they must be stripped off,

```
        PROGRAM MYPRG
```

Also, the comment field is not allowed under UX for functions or subroutines.

4.5. Octal Constants

Octal constants have a different format under UX (standard FORTRAN77) than the typical RTE use. Under RTE, a value followed by the letter B denotes an octal value. UX does not support this, so value must be converted to the standard form. The standard uses the letter O followed by a string of digits in (single) quotes.

The RTE octal constant

```
        I=20040B
```

must be changed under UX to

```
        I=O'20040'
```

Don't overlook the usefulness of the UNIX streams editor, sed. A simple script file for sed can be used to perform bulk modifications on the ported FORTRAN source files.

For example, the following sed script file can be used to edit FORTRAN sources.
groom.sed

```
/FTN/a\
$SHORT \
$HP1000 ARRAYS
/FTN/d
/$CDS ON/d
/PROGRAM/ s/\(PROGRAM .*\)(.*/\1/
/[0-7][0-7]*B/ s/\([0-7][0-7]*\)B/O'\1'/
```

This file will perform the following:

- insert the $SHORT and $HP1000 ARRAYS lines
- delete the FTN77 line
- delete the $CDS ON line
- strip the program type, priority and comment off the PROGRAM statement
- convert octal constants from the B format to the O format

To execute the file run

```
sed -f groom.sed myprog.ftn > myprog.f
```

This will edit myprog.ftn using the groom.sed script and save the edited results in myprog.f.

5. HP-UX CSL Tools

All of these tools can be found on the INTEREX HP-UX Contributed Software Library (CSL) tape, rev. 3321.

5.1. Migration Analysis Utility (MAU)

The Migration Analysis Utility (MAU) is a piece of the PORT/RX product which scans an RTE FORTRAN program and flags each of the calls which are RTE specific. That is, those portions of the RTE program which must be changed to compile and execute under HP-UX.

The number of each type of non-HP-UX call is counted and both the raw counts and the percent of code needing to be changed is reported. These numbers can then be used in estimating the size of the job.

Since many of the RTE routines are supplied in the libraries described below, you should look closely at the MAU results before blindly accepting all the lines of code as needing change. For the most part, the EXEC calls are what will need to be changed. As you'll note below, the generic RTE library routines and most of the FMP routines are available for HP-UX.

5.2. RTE Library (libnat)

Libnat is the RTE library in native HP-UX form. These routines are not emulated, they are implemented using UNIX calls and are therefore very efficient. Most of these routines are documented in the RTE Library manual. Here's a brief sampling of some of the routines contained in this library.

abreg
casefold
charsmatch
clcuc
clearbuffer
cnumd/cnumo
daytime
decimaltoint
elapsedtime
jscom
kcvt
lgbuf
limem
movewords
numerictime
opsys
parse
putincommas
resettimer
sfill
sget
smove
splitstring
sput
timenow
trimlen

As can be seen, most of the more common routines are available.

5.3. File Management Routines (libfmp)

Most of the more common FMP routines are already available in the FMP Library. The library contains the following routines.

FmpBuildName
FmpBuildPath
FmpClose
FmpOpen
FmpParseName
FmpParsePath
FmpPosition
FmpPost
FmpRead
FmpReadString
FmpRewind
FmpRunProgram
FmpSetPosition
FmpSetWorkingDir
FmpSize
FmpWrite
FmpWriteString

5.4. EDIT/1000

There was a version of the EDIT/1000 editor which was supplied with PORT/RX, but it is not in the CSL. That editor, called "ed1000", was really a bunch of macros which sat on top of UNIX's "vi" editor. While the editor was functional, it still had a look and feel more like "vi".

There is an actual version of EDIT/1000 available form a third party for HP-UX called "ed1k". It is Paul W. Miller, Inc.'s PC/EDIT-1000 ported onto HP-UX. This editor has the full look and feel of EDIT/1000, including regular expressions.

5.5. What's Missing?

The items missing are perhaps some of the less frequently used routines. These include the mathematical routines, double integer routines, the VIS routines, and the HpCrt library.

Of course, there isn't any command shell which looks like CI. Perhaps someone will consider implementing "ksh" macros to provide a CI like environment in the future.

6. Possible Future Tools

The Real-Time Advisory Council (RTAC) has been lobbying with HP for several years to provide more help in the migration from RTE to HP-UX or HP-RT. HP has been very interested in finding ways to support the RTE users and their migration to UNIX. Currently several proposals are being considered.

One proposal which the RTAC favors is for HP to provide many of the tools previously mentioned, and a FORTRAN translator. This FORTRAN translator would sanitize the code into acceptable UNIX FORTRAN and replace some of the RTE Calls into an equivalent UNIX code. Perhaps the prime example would be to convert EXEC calls to a UNIX equivalent. An extension to this is to have the FORTRAN converted to C code as well.

References:

Murray W. Hertz, Jr, "Strategy for Migrating and Supporting A Product on Both HP9000/800 and the HP1000", INTEREX Orlando Proceedings, August 1988

Richard M. Pfeiffer & J. Robert Ryan, "Porting A Real-Time Package to HP-UX", INTEREX HP Technical Conference San Jose Proceedings, October 1987

PAPER NUMBER:	1014
TITLE:	MEF Users - Should I Consider an Upgrade?
PRESENTER:	Esther Heller Hewlett-Packard Co. c/o Ella Washington 19091 Pruneridge Ave. M/S 46LG Cupertino, CA 95014 408-447-1053

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Paper 2000

Remote Network Monitoring and Fault Isolation using OpenView

Jeff Hodges
Hewlett-Packard Company
100 Mayfield Avenue M/S 36U0
Mountain View, CA 94043
(415) 691-5525

ABSTRACT

Remote network monitoring and fault isolation for large networks is often a labor-intensive, time-consuming process. Monitoring, problem resolution and reporting tools are required to perform the process. These tools often require considerable operator effort to recognize problems, document the resolution, and generate summary reports. In this paper we will demonstrate how the integration of these tools, using the OpenView platform, can significantly reduce the amount of operator intervention. We will also explore how to enhance the system to provide remote system management.

Introduction

Over the past several years the computer industry has seen a dramatic migration of computer systems, from the "glass house" to the desktop. The power that once resided in the largest of mainframes can now be obtained in a machine the size of a pizza box. With this new, relatively low cost source of compute power has come an increasing dependence on local (LAN) and wide (WAN) area networks. The applications and data that once resided on the company mainframe, in a central location are now distributed throughout several remote sites and are accessed remotely via the network. As companies distribute the applications and data across the geographical topology of the company, the size and complexity of the network grows accordingly. The network that could once be monitored with a pad of yellow post notes and a walk around the computer room now requires a team of network and system engineers located throughout the company and a suite of sophisticated network management applications.

This distribution, and the dramatic increase in size and complexity, pose several new problems to the network engineers responsible for maintaining the company network.

1. With the increase in network size, sophisticated network devices, such as routers, bridges, gateways, hubs and probes, become essential. In addition to the normal system monitoring, these devices require constant monitoring and maintenance, to ensure that the network does not become overloaded. Since these devices are often located remotely, and require special expertise to operate, it would not be cost-effective to have a network engineer at each device location.

2. Since the network is no longer contained at a central site, the network engineers are distributed at key sites throughout it. Problem resolution tools and procedures must be implemented to ensure that the responsible parties work in unison to restore service when an outage occurs. Known problems and resolutions must be electronically accessible to ensure that a network engineer is not spending time working on a problem with a known solution.

3. Due to the sheer number of the devices located on the network, historical data must be collected and trend analysis performed to ensure that the network traffic is appropriately distributed across the network devices. Heavily loaded devices and links must be identified and corrected before failure occurs.

These issues have prompted a large number of software companies to develop software tools to monitor and identify problems with network devices. HP Open View Network Node Manager, SunNet Manager, and IBM NetView are all examples. These tools typically provide mechanisms to collect statistics and generate reports on individual network devices.

Software companies have also targeted the need for problem resolution (trouble ticketing) systems, with products such as, Remedy Action Request System, Networx Paradigm, Quintus SMS, and Peregrine PNMS. These tools all provide mechanisms to allow users to open, assign, document, and close trouble tickets. They provide a way for multiple engineers to work on and solve a given problem, each using their special expertise.

Even with these sophisticated tools, network engineers are often consumed by fire-fighting exercises and cannot accurately document and resolve network events. Consider the following example:

> *A network engineer notices a workstation down when the monitoring tool he is using turns an icon representing the device red. The engineer starts to open a trouble ticket, when another workstation icon turns red. The engineer notices the outage but must continue opening the ticket. Before he can finish, a critical router turns red and his attention is automatically taken from the two workstations and applied to the router. He*

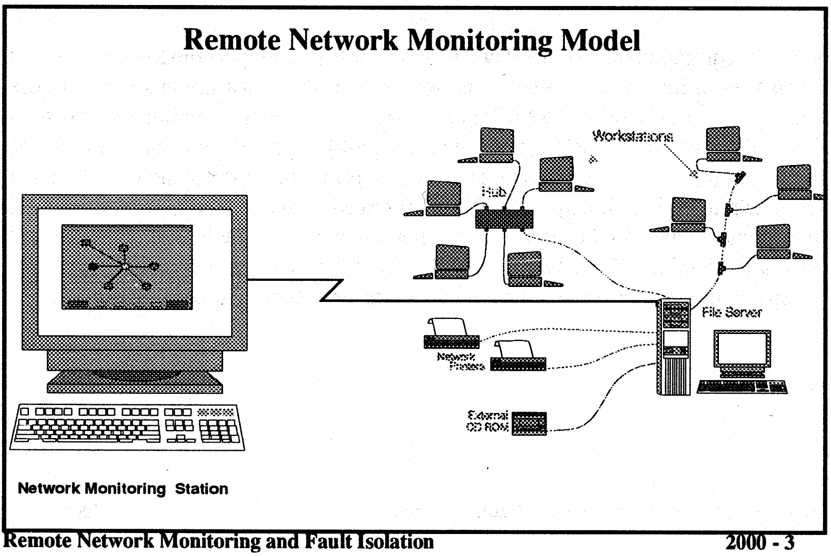
successfully restarts the router and documents the solution. By this time the two workstations have come back up and no action is required.

All seems to be okay: both of the workstations are up, and network traffic is restored through the router. The problem occurs when trying to determine the availability of the workstations. Both of the workstations would not have any explanation as to why they were down during the router outage. This can really be a problem if the workstations have actual defects, and are constantly overlooked due to fire-fighting. In this case, the workstations could fail completely, and there would be no preliminary notification. Worse yet, if a network service provider were responsible for monitoring the network, they would have no explanation as to why the workstations were down and would have a hard time justifying their cost.

In the pages that follow we will construct an environment, using HP OpenView as the base, that will simplify the network engineer's job. When we are through with our construction, we will automatically generate trouble tickets based on qualified network events. We will be able to automatically reconcile the trouble tickets to explain why a given device was down at any time. We will also be able to exclude any outages that may be related to scheduled maintenance. The data collected by the monitoring system and the trouble ticketing system can also be correlated and used to anticipate problems. We will then take our monitoring to the system level and show how a system management tool easily fits into the environment.

Model

For the purposes of our discussion we will use the following system model.



We will assume that we are monitoring a large (>1000 node) TCP/IP based internet, from a remote site located somewhere in the internet. We will assume the network has a number of bridges, routers, hubs, and probes, in addition to the computers and printers.

HP Open View Network Node Manager will be used as the monitoring tool on the monitoring station. In practice, any network management tool with a graphical representation of the network and the ability to create an event log file would suffice.

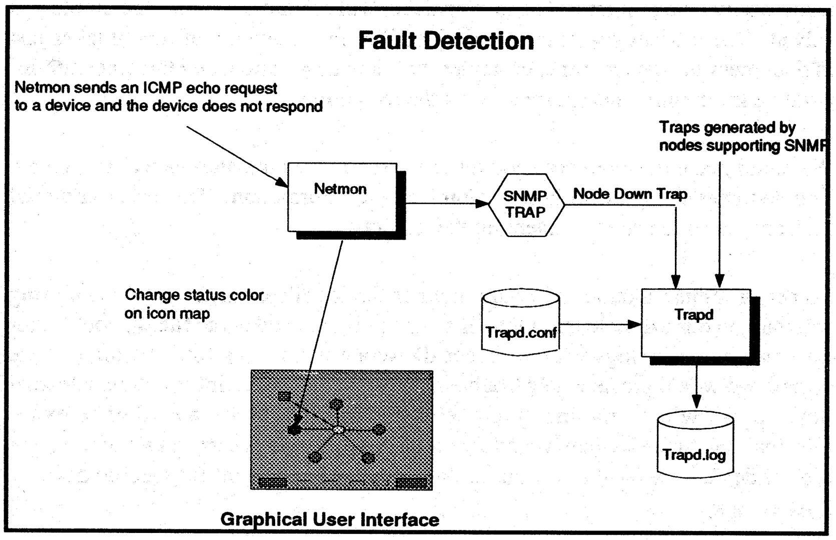
We will assume that network engineers or service providers are dispersed throughout the network, and are coordinated by the network engineer at the monitoring station. Based on these assumptions, we will now construct our environment.

Monitoring Faults

The first item to consider is the network faults (events) you want to monitor for. The most obvious is the status (up/down) of the device. On TCP/IP based networks this can be determined by sending an ICMP echo request (ping) to each device monitored. If the device does not reply within a given time period then the device is considered down. Network Node Manager performs this function via the "netmon" daemon. If "netmon" determines a device is down, it will change the color of the graphical icon representing the device, then generate an SNMP (Simple Network Management Protocol) trap with an event ID specifying it as a "Node Down" event.

It is not required that all of the devices on the network support the SNMP protocol. Open View utilizes it to channel all network events through a single point of access, namely the "trap" daemon. SNMP does make application generation and statistical data collection much easier, given that the protocol is independent of platforms, thus allowing network engineers to access information in the same method on an HP machine as on a Sun system. We do not need to discuss the SNMP protocol in any greater detail for our discussion. Recognizing that each network event can have a unique identifier associated with it, and the fact that it provides a platform independent mechanism to collect and store information, will be enough.

The fault detection diagram illustrates how the Network Node Manager handles network events.


Fault Detection
Netmon sends an ICMP echo request to a device and the device does not respond
Traps generated by nodes supporting SNMP
Netmon
SNMP TRAP
Node Down Trap
Change status color on icon map
Trapd.conf
Trapd
Trapd.log
Graphical User Interface

We may also wish to monitor network devices for heavy utilization. We could establish thresholds for a performance characteristic and then periodically check the characteristic on the network devices from the management station or from a process running on the device. If the statistic exceeds the threshold, we would generate an SNMP trap with an event ID for threshold exceeded.

It's clear that we could extend this principle to monitor non-network information such as critical applications, disk space, and processes.

We now have a tool that will notify the network engineer of the events we wish to monitor. In the next sections we will examine the process that needs to occur when a network engineer is notified of a network event.

Problem Resolution

Due to the size of our network (>1000 nodes), we can expect to have a team of network engineers whose sole function is to keep the network and its devices working. To ensure that the network events detected are resolved without any unnecessary engineering effort, we will use a problem resolution (trouble ticketing) tool. This tool will provide the engineers a way to document the problem resolution process.

Each time a qualified network event is detected, an engineer will open a trouble ticket. The trouble ticket will document the effort performed by the engineering team to solve the problem. The trouble ticket will remain open until the problem is solved. The difficulty with this type of tool lies in the amount of time it takes just to document the device, time, customer, problem description, and the other administrative information necessary to track the resolution.

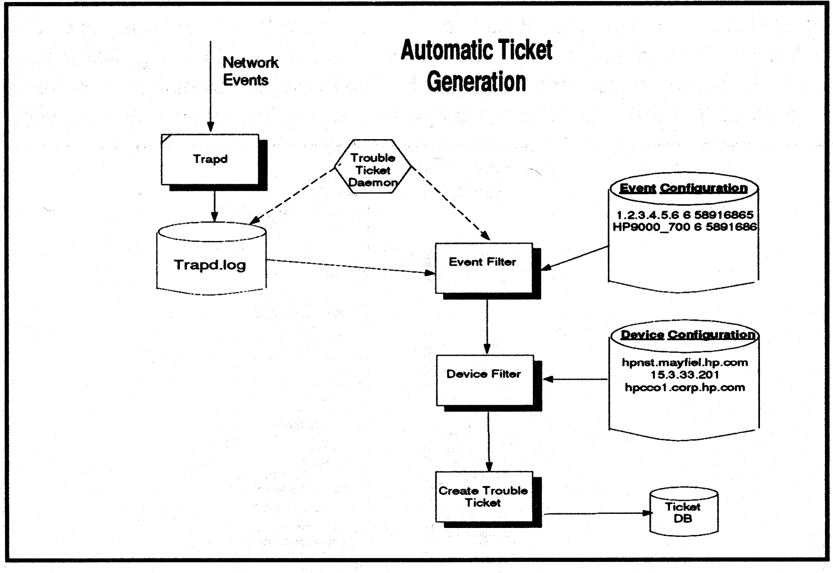
We could make the engineer's job much easier if we could automatically open the trouble ticket and then fill in the administrative information. The engineer would then only need to start documenting the resolution.

To perform this function, we could monitor the log file created by the monitoring software, in our example trapd.log. If a change is detected in the file we could filter the new entries for logs with the event ID we are monitoring for. If a log met the criteria we would create a new trouble ticket with the device information, a textual description which matches the event ID, the time, and any other relevant information. Since we may want to monitor for different events on two of the same type of devices, we will also want to filter each network event for specific devices. For example:

> *Suppose we would like to monitor 3 workstations. Obviously, we would want to monitor them all for up/down status. In addition, one of the workstations is utilized as a gateway and thus we would like to monitor the LAN traffic through it.*

If we only filtered on the event ID we would have trouble tickets generated each time any of the devices went down. However, since we are only filtering on the event ID we would also get a trouble ticket each time any of the devices had a high level of LAN throughput. To overcome this problem, we could filter the log entries a second time for specific devices.

The following diagram illustrates this principle.


Automatic Ticket Generation
Network Events
Trapd
Trouble Ticket Daemon
Trapd.log
Event Filter
Event Configuration
1.2.3.4.5.6 6 58916865
HP9000_700 6 5891686
Device Filter
Device Configuration
hpnst.mayfiel.hp.com
15.3.33.201
hpcco1.corp.hp.com
Create Trouble Ticket
Ticket DB

As can be seen from the illustration, only the logs matching the event ID and device name will cause the generation of a trouble ticket.

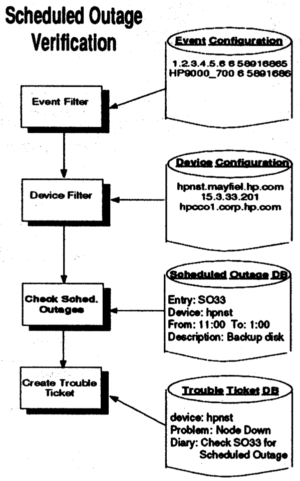
A more sophisticated model would search the trouble ticket database for similar solutions and then invoke a process which would correct the problem. If a match could not be found then the system could search a database of engineer profiles and then assign the trouble ticket to an engineer with the expertise required.

In any of the models we implement, we will have to accommodate scheduled maintenance.

Scheduled Maintenance

In a "ideal world" computers would never need maintenance. Unfortunately, for network and system engineers, the real world is far from ideal. Network devices are routinely being removed from service for upgrades, patches, and repairs.

These outages are normally scheduled such that they have the least amount of impact on the users dependent on the devices. These "scheduled outages" are often cause for false alarm. If the outage is not documented in a manner easily accessible by the network engineers performing the monitoring, a new trouble ticket will be opened and valuable engineering time will be wasted. To prevent this from occurring, we will create a database of scheduled outages. Each time a trouble ticket is created, we will search the scheduled outage database to determine if the device is scheduled to be down. If a match is discovered, we will indicate so in the trouble ticket and the engineer can close the ticket. We will avoid closing tickets automatically to prevent network events unrelated to the scheduled outages from being dismissed. The following diagram integrates this principle into our environment.


Scheduled Outage Verification
Event Filter
Device Filter
Check Sched. Outages
Create Trouble Ticket
Event Configuration
1.2.3.4.5.6 6 58916865
HP9000_700 6 5891686
Device Configuration
hpnst.mayfiel.hp.com
15.3.33.201
hpcco1.corp.hp.com
Scheduled Outage DB
Entry: SO33
Device: hpnst
From: 11:00 To: 1:00
Description: Backup disk
Trouble Ticket DB
device: hpnst
Problem: Node Down
Diary: Check SO33 for Scheduled Outage

One could further simplify the process by automatically loading the scheduled outages periodically, using e-mail or some other electronic mechanism.

We now have a management station which monitors network events, collects device statistics, and automatically creates trouble tickets. We now need a mechanism that will correlate and report the information.

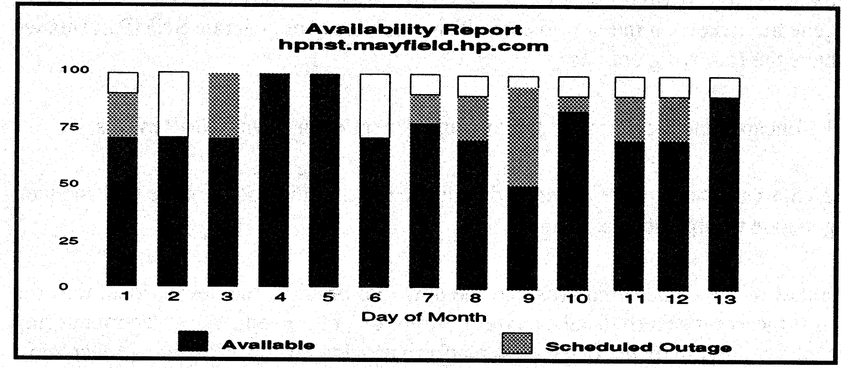
Report Generation

Our environment is now configured to collect and log all network events we are monitoring. Each time a qualified event is detected a trouble ticket is generated. If a scheduled outage is associated with a trouble ticket it is duly noted on the ticket. Using this information,we can generate a number of useful reports.

To simplify the report generation process we will load all of the event logs, statistics, and trouble tickets into a database. This step can be avoided if the trouble ticketing, and monitoring systems all utilized the same data store.

The first and probably most important report we can generate is the device availability report. This report will tell us the percentage of time that a device was reachable via the network during a given time period. We determine availability by computing the total amount of time in the specified time period, then subtract the amount of time spent in outages. An outage can be determined by subtracting the time of a "Node Down" event for a given device, from the next occurring "Node Up" event.

We can enhance this report by indicating which percentage of the outages were scheduled maintenance. For those outages which were not scheduled, we could print the problem description field from the trouble ticket, to indicate why the availability percentage was low. The following is a sample report.



Using the trouble ticket information we can also generate reports on the types of problems certain devices have been experiencing. From this data we can determine the need for hardware, software and training. For example:

> *Suppose we noticed that we were getting a large number of trouble tickets for a particular device, with the problem description: "disk space threshold exceeded". Using this data we could determine the need for a new disk, or justify the purchase of data compression software.*

Since we are now generating the trouble tickets as the events occur, we will have accurate statistics as to the amount of time expired between problem detection and problem resolution From this information we can accurately determine staffing levels and measure the impact of process improvement tools.

Now that we have established a solid management environment, we will examine how we can successfully integrate other tools. In the next section, we will discuss how we can add remote system management capabilities to our environment.

Adding New Tools

As companies migrate to a distributed computing environment, there is a growing concern as to how to manage these distributed computers. A number of companies have developed system management tools to meet this need.

The true test of our environment will be the ability to add new tools and have those tools generate trouble tickets. To achieve this goal we will integrate new tools at the source of our trouble tickets, the event stream. If the new tool generates SNMP events when problems occur, then we could use our existing process without modification. It would be a matter of configuring our trouble ticketing daemon to generate tickets on the new events. If the tool does not generate SNMP events we have the following options:

1.) Encapsulate the output of the tool and generate our own SNMP events.

2.) Start another trouble ticketing daemon to monitor the output of the tool and then generate trouble tickets.

Either of the choices requires that the new tool provide, the device identifier, the time the event occurred and a unique description of the event. All of the monitoring tools, which run on the OpenView platform provide this information. In fact, most of them provide the ability to generate SNMP events and could therefore be integrated with only configuration changes. The following scenario will demonstrate this principle using a system management tool.

> *Suppose we wanted to integrate a system management tool into our model environment. The tool will have a controlling process at the management station which allows operators to send commands to the remote systems. The controlling process will also have a mechanism to receive events from the remote systems when problems develop. On the remote system would be an agent process, responsible for executing the commands sent to it by the controlling process and notifying the controlling process of any problems.*
>
> *If a remote system was having problems the agent process running on it would notify the controlling process, using any communication protocol. The controlling process could then use the same model as the "netmon" daemon. The controlling process would generate an SNMP trap with a unique event id and send it to the "trapd" daemon. The trapd daemon*

> *would log it to the file trapd.log. Our trouble ticketing process would detect this new entry and then filter it. If the entry met the criteria established by the event and device filters then a new trouble ticket would be created.*
>
> *If the tool did not generate SNMP traps then we could have our trouble ticketing daemon monitor the log file it creates. The device filter would remain the same; the event filter would need to use the same identifiers as the system management tool.*

It is clear that our environment can accommodate any network or system management tool that generates a log entries with the device, time, and a unique event ID.

Conclusion

In this paper we have examined how to integrate network monitoring, problem resolution and reporting tools. We have demonstrated how the successful integration of these tools can simplify the network engineer's job. We have also examined how new tools can be added to the environment without significant effort.

Although we used HP OpenView in our model, the architecture discussed could be applied to any network management environment.

Trademarks

HP OpenView is a registered trademark of Hewlett-Packard Company.

NetView is a registered trademark of International Business Machines Corporation.

Action Request System is a trademark of Remedy Corporation.

SunNet Manager is a trademark of Sun Microsystems, Inc.

Paradigm is a trademark of Networx, Inc.

References

Leinwand Allen, Fang Karen, *"Network Management A Practical Perspective", Addison-Wesley, 1993*

Rose, Marshall T., *"The Simple Book",Prentice Hall, 1991*

PAPER # 2001

EVALUATING SYSTEM CAPABILITY:IBM TO HPUX BENCHMARK CASE STUDY

RAYMOND W. RIEDEL
HEWLETT-PACKARD COMPANY
250 N. PATRICK DRIVE SUITE 100
BROOKFIELD, WISCONSIN 53045
(414) 792-8800

Benchmarks are useful for predicting performance with new hardware configurations or software releases. However, no benchmark can exactly duplicate the system being studied. Alternately, performance measurements from the live system are used with analytical modeling to quantify system performance, yet this technique sometimes results in a larger margin for error.This paper will provide HPUX bench marking information as it relates to moving customer specific applications and databases from an IBM environment to a HPUX environment.

A benchmark can be constructed which is representative of the workload on the system. For this, a subset of the configuration's workload is chosen . This may be all the programs if feasible, or just a few that do most of the work (or use most of the resources.) Hopefully , the programs that use most of the resources are doing most of the work. This particular benchmark used a complete Oracle database fully loaded with "live" customer data. The Oracle application topology consisted of the following versions of Oracle:

RDBMS = 6.0.36

FORMS30 = 3.0.15

MENU5 = 5.0.11

REPORT WRITER = 1.1

The application and directory structure was provided by the customer to insure continuity across all the vendor platforms in which they were bench marking. The customer provided scripts necessary to properly build the Oracle system table spaces and the proper term/type files necessary to run their particular application. An Oracle export of the customer's data was provided to HEWLETT-PACKARD to use in rebuilding the application at the Capacity Planning Center. The customer export consumed approximately 2 gigabytes of disk storage plus the storage required to load the necessary modules of Oracle and database log files.

Several tasks should be completed before the benchmark begins. Defining the benchmark with the customer is very important. Developing a methodology for conducting the test is also just as important. In some cases, such as in this one, other systems were used to drive the work flows in conjunction with workload scripts. An actual transaction log from the system could be replayed, or actual users could be called in to perform scripts. The method chosen will depend on many factors, including the availability of testing software(i.e. Empower, etc.), and the amount of resources available for the benchmark and its setup. Fortunately, HP has the available resources to perform such benchmarks if the need arises. Next, assemble a hardware configuration for the benchmark. In some cases use a subset of the full configuration. The goal is to get accurate enough results, without expending too many resources in the process. If only the CPU is being benchmarked, configure the I/O, memory to support this functionality.

What is important is that the final configuration of software and hardware will provide the results that you are seeking.

The customer's benchmark consisted of a mix of tests consisting of retirement related transactions. Postings(deposits, transfers, etc.), inquiries/updates to plans and participants. The customer had informed HEWLETT-PACKARD of the following response time standards for these various tests:

ADP Tests - Each of the people will do one ADP test. The test will be monitored in three pieces. The first will be the setup where the data is entered in to the form. The customer expected sub-second response time in this test. The second would be the calculation of the test (from the point where you press F10 until you are prompted with the "Continue producing report ... (Y/N)". They expected that the response time to be no worse than 1 minute and a maximum of 5 minutes for any of the ADP tests. The third would be the commit. Again, the commit should take no longer than a median of 30 seconds and a maximum of 2 minutes.

Employer Allocations - Each of the people will do one employer allocation. Again, we monitored this in three pieces. The set-up average response time should be less than 1 second, the calculation (from the time you press F7 until the database posting is complete). The customer expected the average response time to be 1 minutes or less and no more than 5 minutes for any of the employer allocations.

The final test consisted of all the "scripted" users doing a combination of the previous tests in random order. The customer expected the response times to be no worse than described in the above tests.

The current application resides on an IBM RS6000 which the customer decided has resulted in less than acceptable response times and currently having little or no upgrade paths. Benchmarks were also done by Sequent using a Symmetry 200/70 processor (10 CPU system at 208MB Ram), and Amdahl using a 5990-700A with 128MB Ram. A NCR benchmark will be arranged after the presentation of the HEWLETT-PACKARD results. The HEWLETT-PACKARD benchmark consisted of a Series 890/400 Corporate Business System consisting of 1 Gbyte of main memory and 6 2.0 GB SCSI disk drives(disk arrays using "striping"). This system was the SUT system(system under test). The system which was used to drive the benchmark scripts on the SUT was a Series 877 with 256 Mbytes of memory and 2 1.3GB SCSI disk drives (See Chart 1).

Chart 1

SUT -------------------------- 2.0GB Array 2.0GBArray

Series 890/400-------------------- 2.0GB Array 2.0GB Array

1GB Main Memory

(DISK STRIPING) 2.0GB Array 2.0GB Array

Both machines were communicating via a Lan

Benchmark Script System

Series 877------------------------ 1.3GB Disk 1.3GB Disk

The customer used a bench marking script development tool called Empower by Performix. This tool allows the user to capture the keystroke sequences of a particular application and replay these scripts to emulate users performing these tasks on a given system in order to determine whether or not the SUT will perform within the given guidelines. Because the customer scripts were generated using this product we were obligated to use the same product to replay the scripts on the HEWLETT-PACKARD hardware. The customer captured the application keystrokes with the Empower product using a HEWLETT-PACKARD Series 700/44 terminal. Therefore, we were able to simply compile the scripts on the Series 877 and replay these script loads on the SUT. The Series 700/44 terminal enabled HEWLETT-PACKARD to perform a couple of key functions during the benchmark. This device allowed HP to insure that access to the Oracle database could be accomplished and that a user could access all the data entry screens of the customer's application. And second, the terminal allowed us to replay the logs from the scripts on the terminal and observe any errors which might have occurred during the execution of any of the benchmark scripts.

The success of this benchmark was dependent on several key factors, some of which were in HP's control and some in the control of the customer. The customer was responsible for capturing the appropriate scripts from their application using the Empower product. They were also responsible for thoroughly defining how they wanted the Oracle database to be configured and how the appropriate data table spaces were to be evenly distributed across the configured disk drives. HP was responsible for understanding the application and scripts to the extent in which they would function properly according to the customer's specifications. Tuning the Oracle application, if necessary, was the responsibility of HP. HP was responsible for compiling the results and presenting them to the customer at the completion of the benchmark.

The representative benchmark is then run, with the important flows being executed. This could involve running them one at a time, as requested by this benchmark, or slowly increasing the workload until the maximum acceptable response time is reached. What this does is allow the person performing the benchmark to determine which resource is being over utilized and at what load level.

The actual preparation and execution of the HP-UX benchmark took place over a course of two weeks. Week one consisted of working with the customer to test load the Oracle database and subsequent scripts on a smaller HP-UX system to insure that the Oracle database could be installed as prescribed by the customer and that the test scripts could be successfully run to completion on a HP system. Without this dry run and preparation, this and many other benchmarks are prone to failure. After a successful dry run, all the data, script files, and appropriate third party application documentation was taken to the HP capacity planning center in Cupertino, California. Upon arrival at the capacity center, one day was spent insuring the proper configuration of the HP-UX operating system and peripherals, and installing the appropriate version of Oracle and applicable products. Oracle 6.0.36 required a few changes to the HP-UX kernel parameters(shared memory parms). The system disks were installed and partitioned using logical volume manager on the Series 890/400. This allowed for one large logical volume to be "striped" for optimum disk performance(spreading the I/O's across several physical disks). Day two was spent completing the Oracle installation and importing the customer's data. As stated earlier, the operation of the customer's application depended upon certain account and directory structures to be in place. These specifications were provided to HP by the customer (See chart 2).

Chart 2

Oracle	/wicapps/oracle
Oracle_Home	/wicapps/oracle
Application	/wicapps/wgora
Forms30Path	/wicmstr/wgora
Menu5Path	/wicmstr/wgora
Database	/wicshare/wgora
Redo Logs	/wicshare/oralogs
Archive Logs	/wicshare/arclogs

To insure that the application was properly installed along with Oracle, the customer was contacted to walk through several of the application screens proving without a doubt that the application and the database was accessible. The 700/44 terminal provided this functionality. These tasks were completed on day three. On day four, the Empower scripts and software were installed on the Series 877 "driver" system. Proper installation of the software in the appropriate account was imperative to the correct compiling and execution of the customer scripts.

HP was provided with 19 separate scripts which were requested to be executed on the SUT(890/400). Each script had to be compiled using the Empower software which in turn generated an executable script. After pre-testing a compile and a run of one of the provided scripts, the full benchmark scenario was started. The request was to have each of the 19 scripts, which were described earlier in the paper, one at a time and collect the results. At the conclusion of those tests, all 19 scripts were to be executed at the same time. Because each of the scripts executed at different rates of speed, a suspend statement was added at the same point in every script, allowing each script to suspend itself at the same point. Using the Empower product, the scripts could all be resumed at the same time providing for continuity in the test. After each completion of a script, an output file is generated. Each output file must then have the data extracted in order to generate a standard report. The customer was interested in the standard report which provides average and maximum response times for each individual script. Observing for any abnormalities during the execution of these scripts is a very important function of this benchmark or any benchmark. For instance, during the execution of a few of these scripts, there appeared to be errors logged to the log file generated during the run. However, the script appeared to run to completion. What had happened was the driver system sent a keyboard sequence to the SUT, the SUT returned a sequence it thought was appropriate. The driver system, however, was expecting something different. Even though the script appeared to complete successfully, it really did not. The scripts for these had to be corrected , recompiled and rerun because of the errors.. Another error occurred where a table could not be found or a field could not be updated. These were problems which were unexpected and had to be corrected with the help of the customer. Provided the resources are available to help in these crisis situations, the benchmark can be back on track in a relatively short period of time. That is why being prepared is VERY important! Arrange to have all the necessary people available in case help is required.

The results of the benchmark were very positive. HP met , and in many cases, exceeded the response time standards set by the customer for this application. All short ADP transactions resulted in sub-second response times and "crunch" calculations resulted in less than the average required response time requirements. During the execution of the full 19 scripts, the CPU utilization was well below 75% with I/O and memory barely being affected. Using disk striping resulted in a significant advantage over "vanilla" independent mode disk drives (See chart 3)

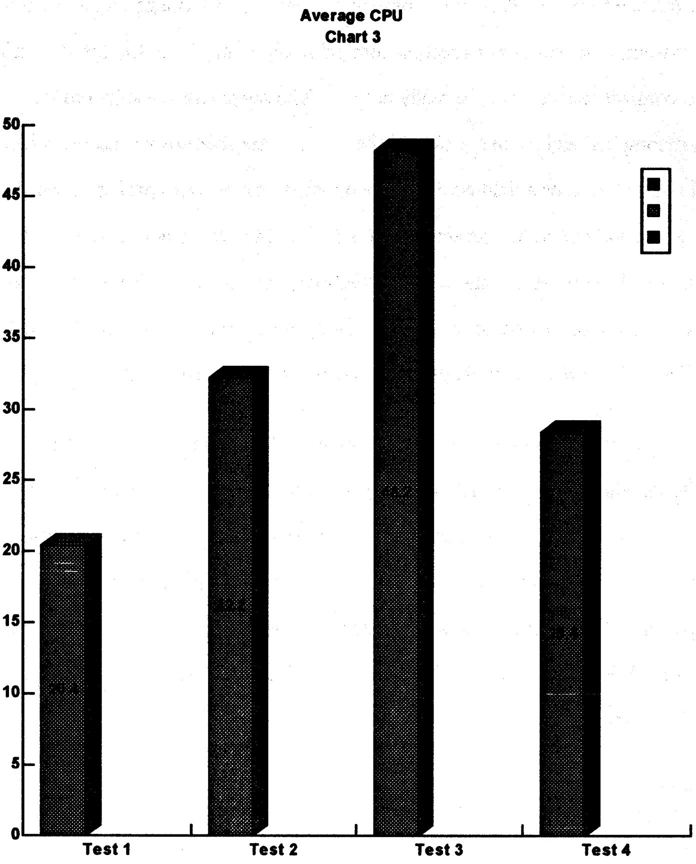

Average CPU
Chart 3
50
45
40
35
30
25
20
15
10
5
0
Test 1
Test 2
Test 3
Test 4

Chart 4

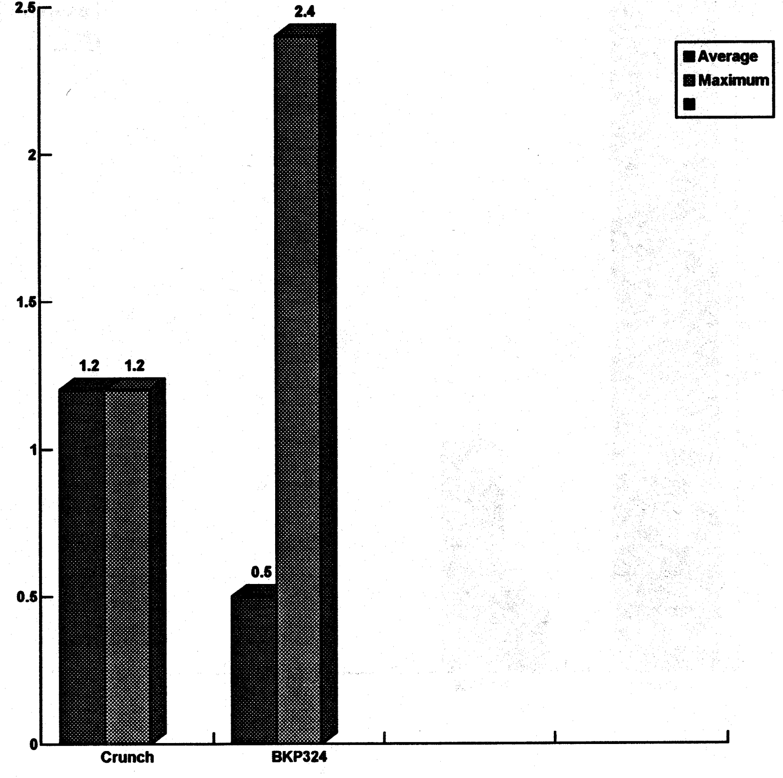


Chart 5

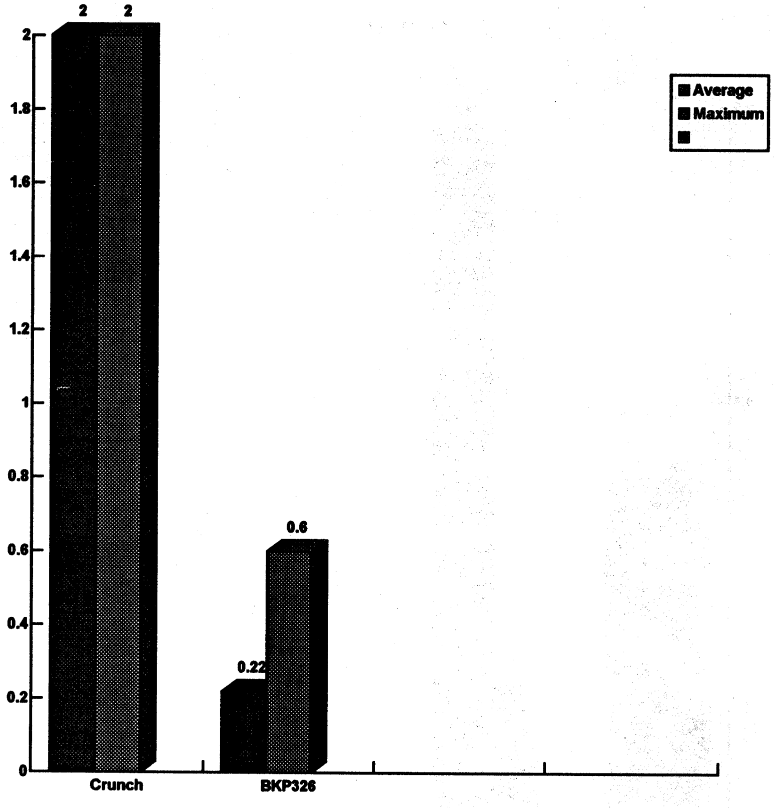

2
2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0.6
0.22
Crunch
BKP326
Average
Maximum

Chart 6

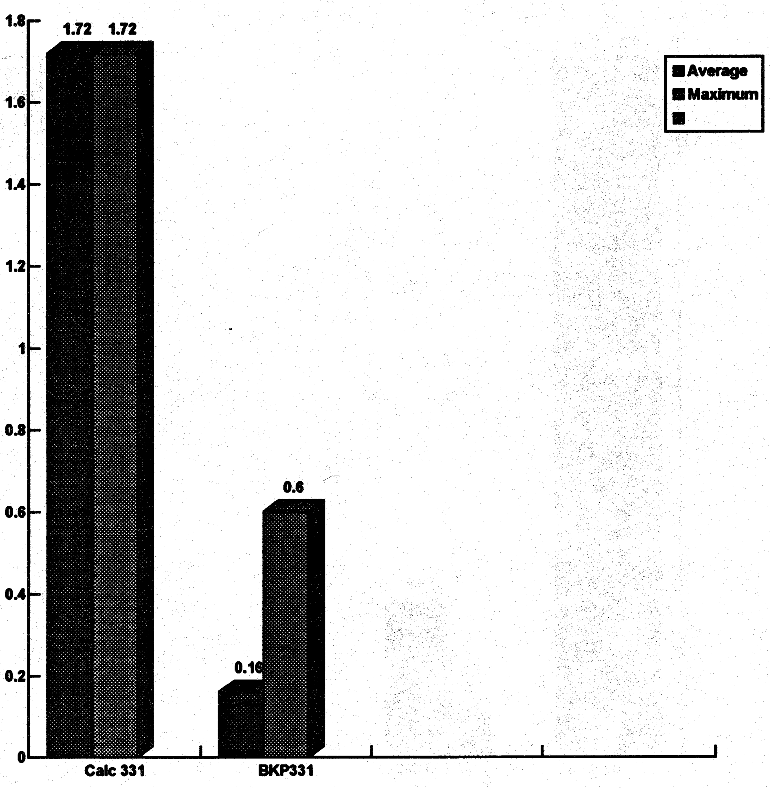

1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1.72
1.72
0.6
0.16
Calc 331
BKP331
Average
Maximum

Chart 7

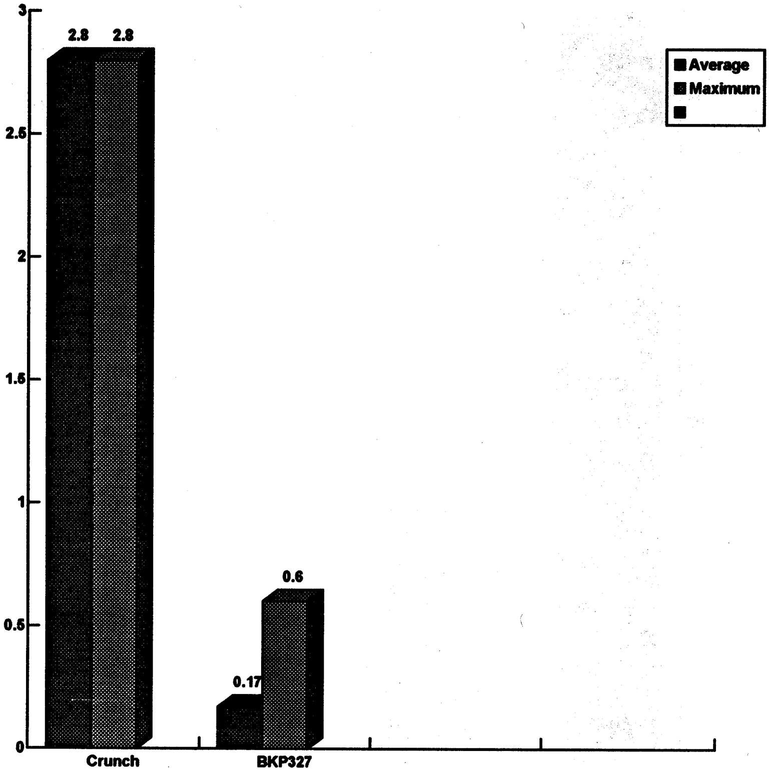
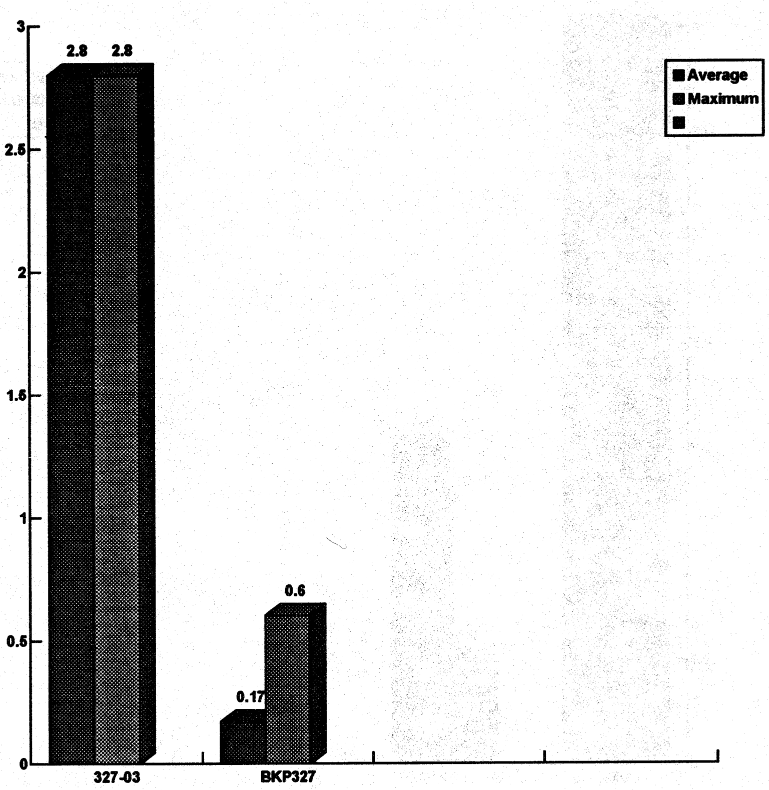


Chart 8



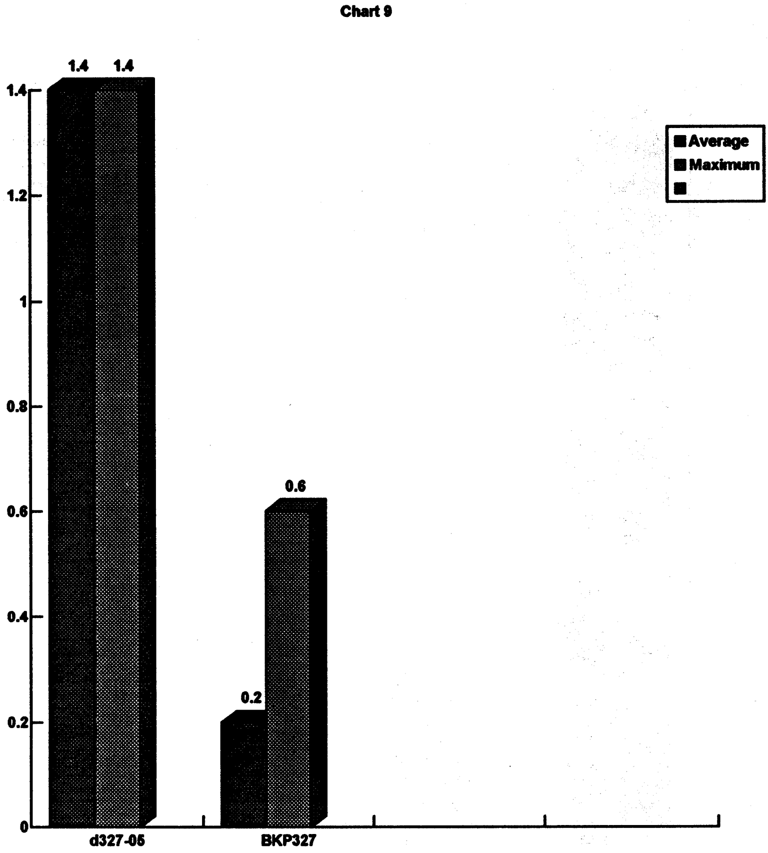


Chart 9
1.4
1.4
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0.6
0.2
d327-05
BKP327
Average
Maximum

There are alternatives to the full benchmark if the situation does not warrant the time and expense of doing a complete benchmark. Another approach would be to use some form of analytic modeling. This procedure involves collecting performance information from the "live" system to quantify the base performance level. This information is used as input to the modeling tool (HP Caplan) to build a base model of the customer's system. This base model is then validated against the measured data to insure accuracy for the task of building future projections or "what-if" scenarios. What-if I increase my data entry personnel 50%? What-if I add another workload with 30 users? What will these changes do to my current response time values? These are some of the tasks that could be accomplished using an analytic modeling tool.. However , given sufficient resources, bench marking is one of the best ways to obtain accurate predictions of future performance, provided also that sufficient planning has taken place. Benchmarks appear to be one of the more accurate ways to determine the effects hardware and software have on performance.

Unlocking the Secrets of UNIX Security

Donna Borsani

The proper administration of UNIX requires a planned, well implemented and systematic approach to issues involving the security of the computer and network, important topics in the plan must include the fundamentals of system protection, data protection, and accountability. System protection is controlling access to the machine, where data protection is the control of access to specific objects, such as directions and files, once admitted. Finally, accountability is the tracing of actions performed by users and activities performed by programs, so you know "who did what when."

The goal of this discussion is to give the system manager insight into the various aspects of security which will enable him to begin security evaluation needs at his/her own site. Details will include discussion about security elements of identification, authentication, authorization and sensitivity levels for people, privileges and sensitivity labels for objects and the differences in the use of discretionary Access Control (DAC) and Mandatory Access Control (MAC).

Additionally the discussion will encompass basic mechanics of network and data transmission, files that control network access, as well as network authentication schemes, such as Kerberos. Lastly we will see how these three fundamental areas of security and their elements are mapped into the different levels of the trusted computer system evaluation criteria, such as C-level (trusted) and B-level (BLS/8.04/8.08).

Paper #2004
Unix Panic? Don't Panic!

Dennis McClure
Hewlett Packard Response Center
2000 South Park Place
Atlanta, Georgia 30339
(404) 850-2742

Abstract

Unix uses the colorful term "panic" to designate an operating system failure, inside the computer. It might also describe what happens to us outside the computer. Our tolerance of computer crashes is lower than ever.

Since porting Unix and developing HP-UX, HP has improved the reliability with hundreds of fixes that prevent panics. These fixes are possible because of diagnostic data gathered from failures. The same diagnostic data from your system's failure can help HP provide you with a fix.

What should you do if your HP9000 has a Unix panic, or some other kind of "crash"? Can you gather the right diagnostic information, or is it more important just to get the system back up and running? You can do both, if you are prepared. None of it is by default, but it is easy when you know what to do.

This paper will discuss coredumps that happen automatically, and how to get one manually. It will discuss the tricky lifecycle of a coredump, from being written originally in the swap area, to being saved in a part of your file system, where it may take up considerable disk space. It will describe the tools used by Response Center engineers to determine the faults, and to find a fix for the problem.

Coredumps can be helpful in pinpointing hardware errors. The paper will cover additional specific procedures for failures in PA-RISC systems.

These functions can be done quickly and calmly in the event of a panic, proving invaluable to the system administrator.

Dennis McClure is a Response Center Engineer in Atlanta, where he has provided help on down systems since 1984. He was a founding member of the HP3000 System Interrupt Team, and is now working on HP9000's.

Panic! Could it be that the HP-UX kernel panics because it does not know what else to do? Yes, although it would be more accurate to say that it panics because there is no better choice. Consider that the kernel has just detected an illogical condition, which the original programmer thinks should never happen. Something is very seriously wrong. The extent of the wrong condition probably goes beyond whatever the kernel detected. Rather than risk writing corrupted data, it is preferable just to stop abruptly. Also, the likelihood of getting a fault fixed is greater if the system stops and provides a diagnostic coredump.

Writing a coredump. An HP-UX panic is one of three events in which a coredump is written, and signifies that software detected something wrong. The fault could be in either software or hardware. A second source of coredumps is an HPMC, or High Priority Machine Check, on an HP9000/700 or 800. The "machine", or hardware, detects the fault in this case, which is always a fault in the hardware. The third source of coredumps is the operator, who can initiate a transfer of control (TOC). A TOC transfers control from the kernel, which is presumably hung, to the hardware's on-board code, which performs the correct series of events for a hang: dump memory, reset hardware, and reboot. TOC's are performed differently on different models of HP9000's.

Most newer 800's, including all the 8x7's, can perform a TOC from the console. Press <control>b, get a CM> prompt, and enter TC. If there is no response to the <control>b, you may have a model with a front panel on which you must first enable the access port or "unlock" the console. This could be done by a keyswitch or a button, depending on the model. Other older 800's have a keyswitch on which the farthest clockwise position is a TOC. This position could be labeled "TC" or "reset". Refer to the owner's manual for your particular model.

Current 700's have a button for this purpose, labeled TOC, TC, or RESET. 300's and 400's do not have a TOC capability, in spite of the reset button on 400's. It just does a reset/reboot. Getting a manual dump on a 300 or 400 requires debug mode.

Where is the dump written? Unless the dump is directed to a specific device, coredumps are written to primary swap. This means the swap space on your root disk, or the swap space that was compiled into the kernel. The swap space is available to write a coredump at the time of the crash, because the swap data there will no longer be needed. On the other hand, sometime after the system reboots, the swap space will be needed again for swapping. Most users prefer letting swap space do double duty as the dump area, but some flexibility is possible.

On 800's, you can direct dumps to a specific hard partition or logical volume using the "dumps on ..." statement. It is used the same way as "swap on ...", in the S800 file. As of 9.0, you can use multiple areas for dumps. See *System Administration Tasks*, Chapter 7, pp. 42-45, for this and other details.

On 700's, you can use the dump statement in the dfile to specify an alternate dump device. Unfortunately, it must be an entire disk, without a file system on it. On the other hand, you can use the swap statement in the dfile to specify that the primary swap space is on a certain disk, and you can specify whether or not that swap space follows a file system. This indirectly puts dumps on the alternate device. With this approach, the minimum disk environment for operation is two disks, the root disk and the primary swap disk. For protection, you might configure /SYSBCKUP to use default swap and dump. See *System Administration Tasks*, Chapter 7, p. 16.

On 300's and 400's, the dump area defaults to primary swap, as on other 9000's. No flexibility for redirecting the dump area is documented. However, the kernel code for swap and dump is the same as on 700's, so the above techniques may be used.

Where does the dump go after rebooting? During the reboot, the /etc/rc script performs a savecore routine. It is conditional; it runs automatically only if a directory exists called /tmp/syscore. Savecore checks to see if the dumps area has a dump in it. If so, savecore copies the dump to /tmp/syscore. This directory is not provided by HP by default, and some users have been unhappy about that. One good reason is because the saving of a coredump can be a serious drain of available disk space. I like the fact that we don't fill up the file system without the system administrator's knowledge of the liability.

The coredump itself will be in a file called hp-core.0. It will be a copy of everything in memory at the time of the crash. The file hp-ux.0 will be a copy of the kernel that was executing at the time. Another file, bounds, keeps track of the numeric suffix that was used, so the next set of files will have ".1" for a suffix.

The savecore program also writes an entry in the /usr/adm/shutdownlog file, telling the reason for the crash, which it cleverly obtains from the kernel message buffer in the dump.

Is disk space available? The coredump file, hp-core.0, will be the size of main memory. The hp-ux.0 file will be about 2 mb. The bounds file is 1 byte. If disk space is limited at either step--in the dump area when the dump is first written, or in the file system when the dump is being saved--the files will truncate. Although partial coredumps are often criticized, they are usually sufficient to characterize what went wrong, and to match with known problems. A truncated dump is insufficient in two problem areas: new problems, and system hangs. In either of these situations, it may be necessary to follow pointers into areas of the dump that are missing.

What if I crashed and /tmp/syscore does not exist? The dump will not be saved automatically, but there is still a good chance to save it manually if you act quickly. In a few situations, possibly with lots of swap space available, and not much need to do swapping, I have saved dumps manually up to a couple of hours after the reboot. The steps are:

```
login as root
# mkdir /tmp/syscore
# savecore /tmp/syscore
```

Can I use a different file system than /? The output from savecore goes to the directory name that is used as a parameter in its execution. If you are running savecore manually, just direct the output to whatever directory you want to use. We do not recommend changing the directory name in the /etc/rc script, however, because this change will be a nuisance to remember and to duplicate when getting a new version of HP-UX. It is easier to leave /etc/rc as provided, but create /tmp/syscore as a symbolic link to the desired disk. For example:

```
# rmdir /tmp/syscore
# mkdir /otherdisk/syscore
# ln -s /otherdisk/syscore /tmp/syscore
```

All references to /tmp/syscore will work ok, but will take place on /otherdisk/syscore.

It is also possible to run savecore with tape output, using the -t option. The dump cannot be analyzed unless it is put back on a disk, but this option does allow saving the dump even if no disk space is available. The tape option is available on HP-UX versions 8.06, 8.07, 9.0 and later.

```
# savecore -t /dev/rmt/0m /dummy-directory-parameter
```

To read the dump back in from tape:

```
# savecore -x -t /dev/rmt/0m /destination-directory
```

Other savecore options. In HP-UX 9.0, a number of changes were made. Savecore will check for enough disk space before the saving the dump, and will skip saving the dump rather than write a partial file. The -p option will allow it to save as much as fits, like it used to do. You can reserve space in the destination directory using a file called "minfree". On 800's, the -i and -k options let you save only the most "important" parts of the dump. See the man pages for these and other options that might be useful in special circumstances.

What about diskless clients? A truly diskless client is a problem for the dump facility, since dumps are written without depending on much intelligence left in the system. The system is crashing at this time, after all. This rules out the use of network software to write the dump on a remote disk drive. Only a "diskless" client that has a local disk for swapping can write a dump. Sometimes if a troublesome situation persists on a client, a local swap disk can be added temporarily to capture a dump.

Reading the dump. Once the dump is safely copied to a file system, what information can you get from it? At this point, I recommend calling for help from the HP Response Center. The steps to obtain information are not difficult, but the interpretation may depend on specialized experience or HP's accumulated data.

Panic message. The first objective is to see an accurate and complete copy of the messages that occurred in the crash. The messages may not have been successfully displayed on the console, or they may not have been observed or recorded by the operator. The following commands will run adb and display the entire kernel message buffer.

```
cd to the directory with the dump files

On 700's and 800's:
# adb -k hp-ux.0 hp-core.0
msgbuf+8/s

On 300/400's:
# adb hp-ux.0 hp-core.0
Msgbuf+8/s

On either, to terminate, enter $q.
```

This is not user-friendly software. Notice that 700's and 800's need a "-k" parameter, and 300/400's use an upper case "M". After executing adb, you may or may not get a line of output that looks like jibberish (but is not). In either case, without waiting for any kind of prompt, enter the message buffer command.

The output will look like what you get from a dmesg command on a running system. It is a circular buffer of a fixed size, causing the space to be reused if necessary. Unless there have been a lot of error messages, you will see every kernel message from booting to panicking and dumping. After boot messages, and before dump messages, there will likely be a couple of lines about the panic, including a set of words telling what type of panic it was.

Stack trace. Of all the numbers, the most useful are the ones following "stack trace". The stack trace addresses may be entered interactively in adb one by one, and the output from each will be the name of the software routine at that address. It will constitute a list of nested procedure calls, in reverse chronological order. See the example below.

Interpretation. The HP Response Center is your best source of information about the things you can see in the dump. You might also search the HP SupportLine database and patches. Sometimes the interpretation will be possible to guess. Here is an example from a 700 (the size of the output has been trimmed significantly):

```
#adb -k hp-ux.0 hp-core.0
u 7FFE6000 u.u_procp 4565D8
msgbuf+8/s

SCSI: bus timeout - bus = 0x1
SCSI: humoring - bus = 0x1, tgt = 0x6, sbcl = 0x22, spc = 0x928
SCSI:   dev: 7201600
        cdb: 28 00 00 20 fa 48 00 00 18 00
        opcode: (28) Read (10)
        status: (400) None -- Incomplete

        Retry count exceeded!

B2352A HP-UX (A.09.01) #3: Fri Dec 18 09:16:31 MST 1992
panic: (display==0xbf00, flags==0x0) pageiodone: B_ERROR set in buffer
```

```
PC-Offset Stack Trace (read across, most recent is 1st):
  0x00125e38 0x0018cda0 0x00191a2c 0x0019148c 0x0013f95c 0x0013eb34
  0x0013e500 0x0013e040 0x001135f8 0x000b35d0 0x00025570
End Of Stack

...

0x125e38
panic+40:
0x18cda0
devswap_pagein+804:
$q
#
```

Note in the SCSI error that the "tgt = 0x6" indicates the device at SCSI target address 6. The device is also identified with a number "7201600", which is a combination of the major and minor numbers. The panic message indicates an io error. The stack trace shows that "panic" was called by "devswap_pagein". Based on this dump, you would correctly guess there was a hardware fault while swapping from the disk at address 6. There are usually some subtleties that are not shown, such as whether the fault is on the disk mech, disk controller, scsi interface, etc, but at least you are on the right track.

One category of panics you can deal with yourself is file system corruption. One of the most common is "panic: free: freeing free frag". This means that the HP-UX routine "free" was called to change the status of a fragment of disk space from "in use" to "free", but the routine found that the frag was already marked free. At the first sign of such illogical trouble, HP-UX will panic rather than risk further corruption. There are a variety of other file system faults, often involving the term "inode" or an abbreviation starting with "i", like "ialloc: dup alloc". An inode was being allocated, and was found to be a duplicate of one already there. File system panics usually have a line of information showing the device address, block number, and the name of the file system.

File system corruption will be fixed by fsck, and there is nothing further to do about it unless it recurs. Recurring file system corruption can be a sign of disk faults. Corruption is also a normal part of any abrupt stop, like a crash or shutting off the power without a shutdown. We depend on fsck to detect and fix things that are wrong. Choosing not to fix something in fsck, by answering "no" to its prompt, is to risk panicking over the fault later.

Dumps other than panics. In a hang or TOC dump, the above technique will not yield good results, unless some coincidental kernel messages have to do with the fault. Likewise, an HPMC dump may have nothing in the message buffer to indicate what happened. A true analysis of the dump is required, using "analyze", HP's dump analysis software. Until recently, this software was not included with customer software distribution. Response Center engineers have had to ask for the dump to be sent in on tape, using a tar relative backup:

```
    # cd /tmp/syscore
    # tar cvf /dev/rmt/0m hp-ux.0 hp-core.0
```

As of 9.0, on 800's only, a version of analyze is distributed called /usr/contrib/bin/scancore. With a dial-in modem, this makes it possible for a Response Center engineer to check out a dump without the delay of shipping a tape. In time, this capability may be available on 700's as well. As indicated by the directory, we do not provide support for you to use it independently. Note that scancore does not have all of analyze's features, so shipping a dump may still be necessary if scancore's output is inconclusive.

Hangs. A dump of a system hang is one of the most difficult dumps to solve. It requires a meticulous analysis of every process on the system, so a full dump is needed. In some cases, multiple TOC dumps are needed to establish patterns between the hangs. As in all cases, your help in describing the circumstances of the dump and helping to isolate the problem will mean faster resolution.

HPMC. A dump of an HPMC can be easy, since the information needed is a concise list of register values. From these values, a Response Center engineer or a Customer Engineer (CE) can determine the failed component. The problem is that certain HPMC's use two registers that are not shown in analyze/scancore output. For that type of HPMC it is necessary to go to the system itself for the data. It is stored in processor independent memory (pim). To display pim, initiate a reboot and interrupt it.

On 800's:

```
Boot from primary?  no
Boot from alternate?  no
Enter command:  pim
```

On 700's:
From the menu, select "a" for boot admin
Enter "pim_info"

The display will include a massive amount of data that is not needed for our purposes, and a few registers that are important. The following is an example of the pertinent part of the display on an 8x7:

```
IIA Space                    = 0x0000000a
IIA Offset                   = 0x00b81b10
Check Type                   = 0x20000000
CPU State                    = 0x9e000004
Cache Check                  = 0x00000000
TLB Check                    = 0x00000000
Bus Check                    = 0x00310000
Assists Check                = 0x00000000
Assist State                 = 0x00000000
System Responder Address     = 0xfffb5000
System Requestor Address     = 0xfffbe000
Path Info                    = 0x00000000
Viper Status Register        = 0x00000102
```

This information should then be reported for interpretation by the Response Center or a CE.

Conclusion. It is bad enough when your system crashes once. It is too much to expect that it might have to crash additional times just to get the right diagnostics. With the above information and a bit of planning, it will be possible to capture, retain, and analyze a coredump the first time. Then, at least, you gain some control over the bad event, and have a chance to prevent its recurrance.

###

Paper Number 2006

Migrating to a Client/Server Computing Architecture One Step at a Time

Patricia A. O'Brien
Client/Server Program Manager
Hewlett-Packard Company
General Systems Division
19111 Pruneridge Avenue, MS 44M6
Cupertino, California 95014
(408) 447-6569

Introduction

Client/server computing has very quickly become one of the most widely used terms in the computer industry. According to a survey taken by the Society for Information Management, client/server is the most important technology to watch, topping electronic data interchange, optical discs and computer-aided software engineering. Forrester Research of Cambridge, Mass. predicts that client/server is the technology that will drive the U.S. computer industry's growth. They believe that client/server system shipments will grow from 28,000 units in 1992 to 183,000 units by 1995 with revenues growing from $4.9 billion to $38.3 billion in those same years. The graph below shows that customers already have moved from investigating client/server computing to running pilots or even full production client/server systems.

HP is already recognized as a major player in the client/server computing market with excellent mainframe and PC connectivity, client and server scalability, a number of client/server application development tools and third-party client/server application solutions. This document will answer these basic questions about client/server computing:

- **What is client/server computing?**
- **Why is client/server computing such a big deal?**
  (or: What benefits will you receive from client/server computing?)
- **Where does client/server computing fit within your organization?**
- **What are the components of a client/server computing solution?**
- **What is HP's strategy for client/server computing?**

What is this concept called Client/Server?

Client/server computing is the computer industry's response to the the business challenges of the decade: global competition, time-to-market and sustained profitability. For businesses to succeed in these competitive times, decisions will have to be made more quickly, product development cycles will have to be shortened and companies will have to become more flexible so that they can respond to their changing business environments. **Client/server offers a model of computing by which information can flow more freely within the organization while retaining the integrity of the data on which that information is based.**

Every vendor, consultant and trade journalist has a definition of client/server computing, but all agree that client/server computing is an architecture and an enabling technology, not a collection of hardware products. In the simplest sense, client/server computing is an environment in which users -- **clients** -- request services from resources on a network of **servers.** The part of the application that resides on the client or on the server should be determined by the application processing environment. Some examples and their benefits are described below.

Ease of Use by adding a GUI to the Client (GUI on Client; Application and Data on Server)

Some customers have implemented a client/server computing environment by adding a GUI to a client that works with legacy applications that run on a mainframe. HP has worked closely with a major oil company that has a variety of mainframe-based applications to collect and process seismic and stratification data and to track site drilling history. The company learned that only 20% of the company's technical staff actually uses the applications that exist. The rest rely on traditional reporting because they don't know where the tools reside and they are unwilling to learn the different access routines required buy each application. HP helped this customer implement a client/server computing model by encapsulating the old applications with a new programming interface that can talk to UNIX clients. The workstation clients now have a consistent interface or GUI across all applications to shorten the end-user learning curve. The result is a uniform way of using the important applications and accessing data without having to wait for a management printout.

Responsiveness to Customers through Easy Access to Information (GUI and Application on Client; Data on Server)

By allowing PC- or workstation-based applications to utilize data that resides elsewhere end-users can be more responsive to their customers. One of the early adopters of client/server computing uses HP 9000 servers to give them access to mainframe data. This telecommunications company's service reps use a client system to access data from the server that pulls customer information from a variety of mainframe-based systems. When a customer calls, the incoming call automatically triggers a display of information about that particular phone number on the service rep's client screen. The service person can immediately answer questions and suggest new services that can turn those complaint calls or questions into sales opportunities.

Quicker Processing when Applications can be run on the Desktop (GUI and Application on Client; Data on Server)

By having applications run an PC's or workstations while sharing data that resides on a file server, companies can reap the benefits of quicker applications processing and improved flexibility. A large bank in the Mid-West has implemented a client/server solution with UNIX workstations and servers. The traders utilize workstations for applications that hook into HP 9000 file servers that hold inventory and position data. Each workstations supports an X-stations that is used for administrative tasks. Applications are being run more efficiently and allow the traders to be more responsive to their customers.

Increased Competitiveness through timely Access to Information (GUI on Client; Application on Server; Data on "Super" Server)

By having access to information at the fingertips of those who need it customers can increase there competitive position in the marketplace. A large aerospace company implemented a three tiered client/server solution and is speeding up product development cycled and reducing processing and communication costs as a result. The company replaced their 3270 terminals with PC's running a Motif-like user interface from Neuron Data. These PC's are clients that request information from HP 9000 servers that are running purchasing, scheduling and MRP applications that previously resided on mainframes. The HP 9000 business server acts as a client to a set of IBM mainframes that hold the data for the applications that are now run on the server. The data is downloaded to the server on a weekly or daily basis, depending on the application -- financial information is downloaded weekly; inventory information, daily. The mainframes continue to run some legacy applications that will move to the server when they need to be revised. Hughes has been able to make a smooth transition from its proprietary host-based environment to a more flexible open client/server environment.

Increased Efficiency and Effectiveness with Client/Server Computing (GUI on Client; Standalone Applications and Data on Clients; Shared Applications and Data on Server)

By connecting "islands of information", sharing resources and making applications easier to user with standard user interfaces, a client/server implementation can help to make employees more productive and better able to respond to change. A large bank holding company has implemented a client/server computing environment with an enterprise-wide network that will connect office process automation workstations, teller workstations, printers terminals and servers from four different vendors. The bank felt pressure to re-engineer their operations because their customers required more information and services. The HP solution at the branch level will provide office automation applications such as OpenMail, Information Access, NewWave Access and HP Office Fax. Applications that reside on mainframe systems will be moved to HP 9000 servers to reduce the cost of processing and to provide easier access to the information. The bank is receiving considerable benefits in the categories of efficiency and effectiveness. Their efficiency benefits include: time savings in processing accounts and loans and reduced communications and printing costs. Effectiveness is increased in the areas of customer service, customer retention and lead-generation.

HP Offers our Customers these Benefits -- and More

These examples illustrate the types of benefits that HP's customers have received by locating parts of the applications where they best serve the need of the organization GUI's running on a client make text-based centralized applications easier to use. Customers with PC's that have had little access to data on other systems can become more productive and more responsive when they are able to share centralized data. These changes don't require customers to completely overhaul their information technology environments. One of the advantages of HP's client/server implementation is that we build a client/server environment based on a customers existing technology infrastructure. This evolutionary approach means that customers can evolve to a true client/server model at their own pace and as their business situations dictate.

The First Steps Toward Client/Server Computing

A client/server architecture may be introduced to your account in the following ways and for the following reasons:

- **adding a Series 800 Business Server to an existing PC-LAN provides better management of shared resources**
- **adding a Series 800 Business Server or a Series 700 workstation as a server in an engineering environment, to a group of standalone clients will give the users the capability to share resources and collaborate on work**
- **adding Series 700 workstations or PC's with a GUI to a mainframe or centralized system makes host applications easier to use**
- **adding Series 800 Business Servers to a mainframe or centralized system can provide faster application processing if an application is offloaded to the server and allows for better and easier access to information that continues to reside on the host**

Adding a Server to a LAN

For small businesses or small workgroups a PC LAN may be sufficient for servicing the file and print sharing needs of the organization or workgroup. In fact today in situations where file sharing and print sharing will remain as the sole function of the server in the long term, PC LANs are likely to be the most cost effective solution.

As the company or user community grows and as cooperative processing across the LAN becomes more prevalent, the PC's will no longer be 'personal', they will become shared resources. In this environment, systems management issues such as backup, software distribution and security becomes critical. A more robust server such as the HP 9000 Series 800 Business Servers may be added to the LAN to perform functions such as backup initiation and to ensure that application versioning issues won't impede the benefits of cooperative processing.

As businesses continue to streamline their business processes, applications supported on a LAN will become more transaction based and as companies flatten their organizational structures the need to distribute information will become more important and may result in the implementation of distributed databases. Both OLTP and distributed database applications require a degree of management, integrity and performance that only a midrange server, such as the Series 800 Business Server will be able to cost effectively provide. Industry consultants at the META GROUP evaluated the role of PC's and midrange systems as servers in a client/server environment and concluded that PC's will not have the systems and network management capabilities comparable to those available on midrange systems today, for at least three more years and PC's will not have the robust capabilities needed for OLTP, distributed database and other mission critical applications for at least 5 more years.

Adding a Server to a Group of Standalone Systems

Another opportunity to introduce client/server computing to your customer is to find groups of professionals, for example in engineering and design areas or desktop publishing, that are using standalone UNIX workstations or PC's. These groups can be made more productive by sharing data and having the ability to collaborate via a network that gives them access to each other's work. Introducing a Series 800 or Series 700 server to store engineering drawings, product specifications or other data that can be shared among the professionals enables your customer to experience the productivity benefits of client/server computing.

A manufacturer of instruments and sensors for commercial and military aircraft uses two Series 800 servers as central file servers providing network-wide design management for over 150 PC's and S/700 workstations. The design tools available for workstations and PC's provide a powerful design environment for the engineers but it creates a problem in managing design data. The Series 800 servers were added so that rather than using a paper drawing, the master file for a design could be stored in one of the servers. The server also performs authorization checks before files are accessed by engineer and routes designs for review and approval at the appropriate stages. A client/server computing solution enabled this company to improve the quality of their products and to decrease their time to market.

Adding Clients to a Mainframe

Most large companies rely heavily on legacy applications residing on mainframes. These applications are usually difficult for users to access and use. Because so many of these applications are mission critical, companies are reluctant to offload them to less costly and more open platforms. HP can still add value to the applications by providing a graphical front-end that runs on a workstation or X-stations like the HP 9000 Apollo Series 700 workstations, Series 700/RX-terminals, and HP Vectra PC's as described in the oil company example earlier. The end-users are more likely to use the application, become more productive with it sooner and make fewer mistakes when a graphical user interface is added to the application.

Adding Servers to a Mainframe

Series 800 Business Servers can be added as front-end or back-end servers to a mainframe that is running mission critical applications. They can also be used to offload mainframe applications altogether. In both cases the S/800 servers make data more accessible and more usable to those who need it. While acting as a client to the mainframe, the Series 800 is a server to the end-users who access the data via PC's, MAC's or UNIX workstations. A large bank in Tokyo has implemented a top-down client/server computing solution. Their mainframe system is the repository for customer information and other information that they collect, but now the applications the the bank's managers need to make use of the data now run on multiple Series 800 servers that are accessed via X-terminals. Now applications reside closer to the people who need them, permitting them to share the same information but apply it in a way that satisfies their customers' requirements.

Why is Client/Server such a big deal?

Client/server computing offers significant benefits to end-users and MIS departments making both groups more productive while utilizing IT resources in a cost effective manner.

Benefits For the End-User - Freedom from Technology

Client/server computing gives organizations freedom from technology. In the past, the development of business processes were sometimes constrained by a company's existing technology infrastructure and capabilities. End-users, instead of defining business processes with their own work-style in mind, were forced to conform to the capabilities of their centralized computing environments. Today, client/server computing frees them from those constraints. By giving the end-user ownership of processing power (through an intelligent client on their desk) and a link to data and/or applications that are managed centrally (to ensure integrity and performance levels) technology is now able to accommodate the end-user's work style and the business processes that the end-user has defined.

Advantages of C/S	Productivity Benefit/Cost Reduction
Easier/Better Access to Data Many of the client/server tools allow access to data that resides on multiple and incompatible platforms.	End-users can now have transparent access to data. They don't need to know where on the network the data reside. Users can now spend more time making use of information rather than trying to find it -- or worse yet, making decisions without it. Better decisions are made when information is available to those who need it.
Ease of Use In a client/server environment graphical user interfaces can be added to legacy applications and come standard on new applications.	Front-ending applications and applications and application development tools with consistent graphical user interfaces can cut down or eliminate training time and expense, reducing costs and enabling users to become productive more quickly. In addition the use of GUI's can lead to an increased number of completed tasks.
Faster Response Time The client/server computing model allows for the optimal placement of applications and data.	Intelligent desktop devices can be used to offload applications that had previously run on host systems, resulting in faster response times for the offloaded applications since they are no longer vying for the same resources demanded by other applications and

users. The host system is freed up for faster processing of the batch or legacy applications that still reside there. Users of both systems will no longer experience and become frustrated by the wait time typically associated with centralized applications. In addition, they can continue their work with local applications during host downtime.

Benefits for the MIS Department -- Better Applications Faster

Today's business climate finds most MIS managers facing a flat or decreased budget. Yet the demands from their customers -- the end users -- haven't stopped and in fact, now that end-users have had PC's and their own applications running on their desks, it is likely that their demands for access to centralized data and increased functionality from the applications the MIS department provides have probably increased. A client/server computing model, built on the existing IT structure, can help MIS to implement change quickly and thus be more responsive to the needs of the MIS customer.

Advantages of C/S	Productivity Benefit/Cost Reduction
Modularity of Applications Application development time can be greatly reduced, more code is reusable and applications are easier to maintain to the modularity of client/server application code.	Because client server code is designed to be modular, development teams can proceed in parallel -- one group working on the client, the other on the server. Modularity of code allows for specialization of programmer skills (some are better at writing GUI's others data access due routines.) More code is reusable because one may choose to develop a server for multiple applications. In general applications for a client/server environment are simpler than traditional mainframe applications where one large application performs a variety of functions. The 4GL's and CASE tools available for client/server environments also simplify application development. Maintaining and updating applications are easier because functions are segmented between the clients and servers. If an application and GUI are running on a client and an end-user wants a new capability in the user interface or application logic, instead of that application change affecting the entire application and the entire user-base, the change can take place on the client for just those end-users that need it.
Centralized Control With a client/server model the MIS department can reclaim control over user PC's and workstations.	Client/server computing helps to ensure the integrity of data. The model separates the data from the applications sparing the IT group from keeping copies of the same data on several different systems. The server also has the rules for how to access and interact with the data ensuring that, although it may be available to a wide range of end-users, it can only be changed or updated by the appropriate individuals. In general client/server computing helps IT to reconcile the divergent needs for

responsiveness to the community while ensuring data integrity through MIS control. In a client server model, the server governs data integrity and access with rules imposed by the IT community, but because the interfaces between client and server are clearly defined, anyone can write and application to run against the server.

Ability to Grow Incrementally
The client/server computing model allows for the optimal placement of applications and data.

Both clients and servers are available in a range of sizes, strengths and prices, enabling customers to tailor their client/server relationships to meet their needs. Previously under-used processing power can be exploited with GUI front-ends and applications that are made more useful by being able to access enterprise data that may reside on a server. As client applications grow they are not limited by existing memory and processing power because additional processing power and large storage are available through the server.

What's Included in a Client/Server Solution?

A client/server computing is not an add-on feature or a specific product solution. It is an approach to computing that can be used to define and describe a customer's computing environment. Components of a client/server computing solution include:

- clients and servers
- networking
- graphical user interfaces
- applications
- application development tools
- application integration tools
- databases

HP's Strategy For Client/Server Computing

HP's strategy is to provide a set of tools and solutions that make it possible for customers to move from their present computing environment to a client/server computing environment at their own pace, as dictated by their business needs. HP's emphasis on a standards-based, Open Systems environment enables customers to leverage their existing information technology investment and ensures that any new investment will be compatible with the technical advances of the future. In addition to offering a complete client/server computing solution, HP has broadened our definition of client/server computing to include the following characteristics.

Open Technologies

HP's client/server computing solutions encompass a wide variety of open technologies based on industry and defacto standards to accomplish a given task and deliver solutions relevant to the business task at hand. For example, the HP model blends the full complement of technologies needed for the creation and delivery of client/server solutions such as advanced GUI's, networking technology, object programming and computing and 4GL's.

What sets HP apart is that we recognize the need for "middleware": DCE (Distributed Computing Environment) and DME (Distributed Management Environment) components such as remote procedure calls, license management services and software that are essential for integrating and supporting distributed applications. (Remember, by definition, client/server means that applications will be distributed across more than one platform.) HP has taken a leadership role in defining these standards exemplified by the seven HP technologies that have been chosen as core components of DCE and DME standards. HP is delivering client/server technologies today with OpenView, Network Nodal Manager, Network Computing Manager and Network License Server.

Legacy Preservation

HP's client/server model does not require your customers to throw out everything they already have. We are not like some vendors who offer only a piece of a client/server solution, or like those who offer to provide a total client/server solution -- but only if the customer is willing to start from scratch. HP recognizes that our customers already have a significant investment in information technology. Our client/server models build on that investment by redefining where and how the processing of information will take place -- that is closer to the people who need the information.

Added Value

Finally HP's client/server model is committed to adding real value to the existing investment by improving user interfaces which enable faster learning. Our solutions provide easier access to existing data and applications while maintaining data integrity. Our model enables the integration of data from a variety of sources, many of which may be incompatible with each other, and transforms that data into useful information. Our CASE offerings provide fast prototyping of client/server applications and our network and systems management environment enables fast and accurate deployment of client/server systems.

HP's Clients and Servers

The HP 9000 Series 800 Business Server

Midrange systems will play an important and growing role as companies move toward client/server architectures. Companies are streamlining their business processes and moving from batch-oriented to transaction-based applications. The distribution of information is becoming critical as companies flatten their organizational structures. As companies move to client/server architectures to support these business and information processing changes, midrange systems play a critical role. OLTP and distributed database applications require a degree of management, integrity, performance scalability and expandability that only midrange platforms can cost-effectively provide. The robust systems and network management capabilities of midrange systems also make them an ideal choice as systems and network management platforms.

The HP 9000 Series 800 is HP's strategic Unix server. The S/800 offers the broadest range of performance, I/O expandability, memory capacity and systems and network management capabilities in the industry. It offers many price/performance points with excellent upgrade paths in addition to future midrange SMP to match the competition. The S/800's fast high capacity networking, server packaging, commercial functionality and applications and tools availability give the Series 800 a substantial edge over the competition.

A Choice of Clients

One of the benefits of client/server computing is to make better use of "sleeping" desktop MIPS. Many of your customers already have PC's, MAC's or UNIX workstations. Many times these desktop these potential clients are not utilized to their full capacity. For example, when a PC enters terminal emulation mode to access a mainframe the user loses all the advantages of having an intelligent device on the desktop. Clients such as PC's. MAC's and UNIX workstations can be made much more productive by pulling parts of an application from a larger, centralized system to the desktop processors. Application processing time will be much quicker when applications are run on a dedicated system, such as the single user PC or workstation would be in this case. Standalone desktop applications can also become more useful when they are able to access and share data.

HP 9000 servers have a unmatched desktop integration story. With partners like Novell, Microsoft and Pacer Software providing Netware, LAN Manager and Pacer respectively, the HP 9000 is able to support DOS, OS/2 and Windows PC's as well as MAC's, X-Stations and UNIX workstations. The networking products that support these desktops also support standard API's for developing client/server applications for these intelligent desktop clients. Thus HP provides all the tools and connections to put your customer's sleeping MIPs back to work.

The Series 700 and X-Terminals as Clients

For customers implementing a client/server solution where no desktop computing currently exists, or for customers who are looking for a more versatile and powerful client, the HP Series 700 workstations and HP 700/RX X-terminals are ideal clients.

HP's Strategic Clients and Servers -- Together

The combination of Series 800 servers and Series 700's and 700/RX's as UNIX clients provides your customer with an ideal client/server implementation. The Unix operating system is the most mature, robust open operating environment that is supported from the desktop to datacenter-class systems and the HP PA-RISC architecture is the only RISC offering that is supported from the desktop to the data center. The homogeneity of of HP's RISC/Unix client/server environment means that customers can spend more time on application development and less time on connectivity, interoperability and other networking and systems management issues. HP's broad, scalable, binary compatible product line means that customers can implement the client/server model no matter how large or small their computing needs and will be ensured that it has the capability to grow with (or be reorganized with) their organization.

Other Advantages

In addition to their absolute performance advantage, UNIX workstations have significant advantages over PC's in the areas of network and systems management. PC's require special servers for even the simplest of administration tools that are standard on workstations. Workstations are based on the concept of workgroup productivity and come with the tools that allow workgroups to be managed more easily. An administrator can update all workstations with the latest version of software from the administrators workstations, rather than plugging floppies into each PC. An administrator can diagnose and fix problems on other workstations even while the user may be performing other tasks on the system, by logging into the users' system from remote or local servers rather than physically working on the user's PC.

Although PC's cost less on the low end, when customers add memory, additional disk capacity, networking hardware and software and high resolution displays they are well into the range of workstation prices. HP Series 700 workstation prices start at near $5,000 for grayscale and under $10,000 for color. The cost-per-seat can be even lower by combining S/700's with HP 700/RX X stations. X-stations offer the lowest price solution for adding multitasking, graphical user interface seats to a network of workstations and servers. They are ideal for users who can rely on an server for storage and processing. HP X-stations can even be easily upgraded to workstations protecting the customers investment.

The speed and power of UNIX workstations make them better equipped than PC's for the demands of real-time applications and most of today's leading PC applications are available on UNIX workstations. As new applications require graphics, multimedia and voice annotation capabilities greater amounts of memory and performance will be required. Workstations are best suited too handle the system requirements of these newer applications.

Graphical User Interface

Graphical user interfaces (GUI's) provide users with a consistent, easy-to-use interface across all client applications which decreases the amount of time end-users must spend learning new applications, thus helping end users to be more productive. Studies have shown that the use of GUI's have offered productivity benefits by increasing the number of applications end users actually use. One study reflects a 35% increase in the number of tasks completed and a 16% decrease in errors. Fifty one percent of those studied experienced less frustrations and 23% attempted more tasks because of the "friendlier" environment. HP supports a rich assortment of GUI's including:

- **HP VUE (based on the OSF/Motif and X Windows standards)**
- **MS Windows**
- **HP NewWave**
- **Presentation Manager**

Networking

Because all transactions pass over the network, client/server computing is not possible without a network connecting all clients and servers. Networking for client/server has many aspects including LAN and WAN networking standards, interfaces, integration, and network management. HP has a comprehensive networking offering that helps customers protect their existing investment while enabling them to take advantage of newer, more open technologies such as client/server computing. HP provides seamless integration to proprietary legacy systems as well as to open systems by complementing our own networking technologies with leading industry technologies. HP recognizes that customers need their distributed environments to be secure and manageable and we have enhanced our networking offerings with the management tools that provide these capabilities. HP supports the following network components that are critical in client/server environments.

Networking Foundations: HP's networking foundation products provide the infrastructure for Local and Wide Area Communications (LAN's and WAN's). They also enable the HP 9000 to integrate into enterprise-wide networks. By providing such a foundation, HP shields the end-user from the complexities of how the network is physically connected. The HP 9000 supports:

Token Ring FDDI Ethernet
SNA/X.25 SNA/SDLC SNA/Token Ring

Desktop Integration: These products enable the HP 9000 to serve clients such as terminals, PC's, MAC's and UNIX workstations and to integrate your customers workgroups into their enterprise. The choice of clients and NOS's (Network Operating Systems) further demonstrates HP's capability to integrate into your customer's existing environment.

LAN Manager/X Netware for Unix ARPA AFS
Pacer Software OSF/DCE NFS Appletalk Banyan

Application Programming Interfaces (API's): Application programming interfaces provide developers of client/server applications with access to the capabilities offered by the network. The following API's enable two-way communications between clients and servers (or between servers and servers) and offer the capability for applications running on UNIX clients and servers to be divided into sections so that each section can be processed by the next available server on the network.

DCE (NCS) RPC TCP/IP: Berkeley Sockets Named Pipes
OSI SPX/IPC (Novell) CSI (Apple API) SNA:LU6.2, HLLAPI

Network Management: HP offers industry leading manageability with products like HP OpenView which has been accepted as a key component of OSF's DME and has been OEM'ed by IBM for their RS/6000. The OpenView product enables one to manage multivendor networks from a single point. There are currently 110 solution partners who have written management applications (such as fault tracking, accounting and billing) to this environment.

DME/OpenView OSF DCE Security (Kerberos) Omniback
OpenSpool PerfView

Database Management

Fast and easy end-user access to data, while preserving the integrity of data is one of the key differentiators of a client/server solution. The database management component of a client/server solution assists in offering client/server users this benefit. The placement of the database management component determines if a system acts as a client or as a server.

The HP 9000 as a Data Server

The S/800's and S/700's are frequently used as a data management servers, meaning that the data and the database engine reside on the S/800 or S/700. (Part or all of the application may also reside there.) The data residing on the server can be accessed by a variety of clients via "NET" products offered by most database vendors. NET products facilitate client/server communications that allow applications on a client to interface with data on the server. Informix-NET is an example of a NET product.

The HP 9000 in a Distributed Database Environment

Parts of a distributed database may run on a Series 700 and/or 800. In this scenario the application processing split takes place within the data management layer. Data may be distributed across a variety of clients and servers. Database vendors' "STAR" products facilitate distributed computing by enabling communications between databases that reside on multiple servers and clients. Most of the leading database vendors have this capability today, but in most cases only homogeneous database environments are supported (i.e. when the databases on each of the systems is the same: all Ingres, all Oracle etc.). When a database is distributed on 700's and 800's each system can be a server and client to each other depending on where the requested data resides.

The HP 9000 as a Client to a Mainframe Database Server

Finally, a Series 800 Business Server also has the capability of accessing data that resides on mainframe systems. Using SQL access products such as EDA/SQL (Enterprise Data Access) from Information Builders and "Gateway" products that are available from Oracle, Ingres. Sybase, Informix and Cincom, Series 800 servers can act as clients requesting data from a mainframe database server with IBM's DB2, IMS and SQL/DS or CA's IDMS and Datacom databases. In turn the same Series 800 acts as a server to workstations or PC's that request data from the Series 800. The end-user does not need to know if the data is actually stored on the HP system or on one of multiple mainframes.

In each of these scenarios, it is important for the RDBMS to be tightly tuned with the server. HP's strong relationships with the leading relational database vendors ensure such tuning for the HP 9000. Other database features especially important for client/server computing include:

- o **Rule integrity**, whereby business rules can be predefined and "triggers" ensure that transactions against the database do not circumvent the rules
- o **Locking**, which allows client users to behave as if they have sole database access; most database vendors support this at a page or row level
- o **Distributed query management and optimization** for most efficient use of the distributed database
- o **Two-phase commit functionality**, so that transactions won't be committed unless all systems and databases involved are ready for the transaction
- o **Data access** between clients and servers and from heterogeneous databases

HP's database management partners who achieve client/server via combinations of the above features are the following:

Cincom
Informix
Ingres
Oracle
Progress
Software AG
Sybase
Unify

Application Software

Application software vendors are answering customers demands for client/server implementation of their software packages although the vendors have taken different approaches to offering this functionality. Many application software vendors have enhanced their standard product with a GUI that runs on a PC or UNIX workstation, offering customers the ability to take advantage of desktop MIPs. Others have chosen to optimize the processing of their applications by splitting the application logic across clients and servers sometimes giving customers the choice of where processing modules will run. Finally, a number of vendors have the ability for the database to reside on a different platform than the application, perhaps on a mainframe or centralized server for broad end-user access. SAP, Peoplesoft and Dun and Bradstreet are some of the leaders in reengineering or developing their applications to a client/server model.

Application Integration and Development

Developing applications for a client/server environment differs in several ways from traditional application development. Design tools and code generators must generate code for a server and a client. Usability becomes more important as GUI's give developers many more choices for application presentation than character-based design. Optimizing the application processing split -- choosing what parts of the application should run on the server vs. client and allowing for future flexibility add complexity to the application development process.

The languages, standalone tools, integration framework and integrated CASE tools that HP offers as our CASEdge strategy reduces the complexity of developing in a client/server environment.

Languages

Client/server applications can be developed by using languages products such as COBOL , C and C++. These are usually a lower cost alternative to 4GL's or CASE tools. They can be based on in-house expertise and tend to be more "open" because they are defacto development standards.

Standalone Tools

HP has been successful in recruiting the market leaders in standalone application development tools. These tools can be integrated with upper-CASE tools and into open integration frameworks such HP's Softbench. Although the tools are excellent for rapid prototyping applications, they should not be considered a substitute for complete lifecycle planning.

Look for tools in the following categories:

- o **4GL's and screen painters without GUI support**
  - used for building terminal style user interfaces
- o **4GL's and screen printers with partial GUI support**
  - for building mixed terminal and GUI interfaces
- o **Full GUI client/server development tools**
  - used strictly for GUI development with a 4GL
- o **Frontware**
  - used for adding a GUI to an existing application
- o **User interface management systems**
  - used for database independent GUI development

Open Integration Framework - Softbench

The second product area within the CASEdge strategy includes tools that are loosely integrated together within and open framework. HP Softbench provides such a framework and achieves integration by "encapsulating" tools, giving them a common user interface and enabling them to pass messages to and from one another. This category provides a higher level of structure than stand-alone tools for larger, more complex projects.

Softbench runs on the HP 9000's, Apollo and Sun platforms today and is being ported to additional platforms. IBM has licensed it for use with their RS/6000's. They call it WorkBench. Informix has selected it to be the framework technology for its OpenCASE Toolbus product, and CDC has licensed it to be used with their MIP's-based systems.

Integrated CASE

Integrated CASE, or i-CASE, provides full lifecycle, structured CASE environments for large-scale, complex development. HP offers i-CASE products, previously available only on mainframes, of these leading vendors who have chosen the HP 9000 as their break into the commercial UNIX market. Each of these vendors (especially Andersen and TI) have been recognized for their client/server development functionality.

Andersen:	Foundation for Cooperative Processing
CGI:	PacBase, PAC/Lan, PAC/Lan/X
Softlab:	Maestro II
TI:	IEF

Application Integration

Two HP products that enable diverse applications to share data and results are NewWave's Object Management Facility (OMF) and HP Sockets. OMF allows workgroup members to share and combine different kinds of files in a single document "object". Data within the object remains consistent not matter what the format. For example, if data in the spreadsheet or graphic is changed, it is automatically changed wherever else it appears in the word processing text of the object. HP Sockets links applications in a Computer Integrated Manufacturing (CIM) environment allowing manufacturing customers to integrate new applications with existing applications on multivendor computer systems.

The HP Client/Server Advantage

HP's client/server solutions offers significant advantages including:

- **More effective decision-making because users can get the information they need when they need it.**
- **Flexibility to respond to marketplace demands because users can access information faster.**
- **Increased productivity throughout the organization due to standard interfaces to applications and tools.**
- **Cost reduction due to the ability to take advantage of desktop compute power and to optimize the placement of applications and data.**

To help our customers to achieve these benefits HP offers some advantages that differentiate us from our competitors.

Single Vendor Solution/Single Operating Environment

HP's comprehensive client/server solution based on a single hardware architecture and operating environment offers advantages unmatched by our competition. A single operating environment allows for a single application version easing support requirements and training. Integration of tasks between a client and server is easier because one may use common tools and utilities. Managing the systems and network is done more efficiently for the same reason. A single vendor solution also allows customers to spend less time managing vendor relationships and more time managing their business.

HP's Scalable Systems

The range of performance of HP's clients and servers is broader than that of our competitors. This scalability is especially important in client/server computing environments because demand for compute power grows once users experience capabilities like better access to data and applications that are easier to learn and use. HP's scalable systems can grow along with your customers needs. Scalability from the desktop to the datacenter also gives HP the ability to help our customers experience the benefits of client/server computing anywhere within their organization.

HP's Networking Protects Current Technology Investment

HP's robust and comprehensive networking offerings and strong commitment to industry standards enables us to offer client/server capabilities that build on customers' existing IT infrastructures, allowing them to move to client/server at their own pace while retaining and enhancing their past investments. HP's networking offerings allow customers to to turn their desktop systems into clients and mainframe computers into servers making their their existing technologies more productive.

HP: a Technology Leader in "Middleware"

A distributed, networked environment is a characteristic of a client/server environment. HP is an industry leader in enabling customers to manage this new environment. The Distributed Management Environment (DME) will be the standard for managing networks of distributed systems. HP provides customers with the fastest and smoothest path to DME and addresses customer needs with products available today. HP OpenView Network Management Server, HP Network License System (NetLS) and HP Software Distribution Utilities were selected as key components of the DME, reaffirming HP's focus, commitment and leadership in open systems and in the management of distributed environments.

HP Partners with the Leaders

HP has developed partnerships with the leaders in providing applications, databases and tools for the client/server environment. Vendors like Sybase, Ingres, Dun & Bradstreet and Computer Associates all recognize the advantages and growth associated with client/server. They also recognize that HP is positioned well to make the most of the opportunity. HP has made client/server solutions available than any other UNIX vendor.

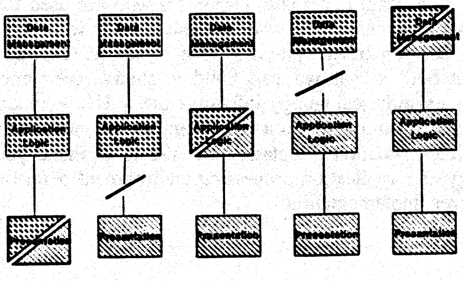
HP has Client/Server Experience
HP has experience implementing client/server solutions. Our Professional Services Organization and systems integration partners have already provided a number of customers with a smooth transition to a responsive client/server model of computing. In fact, HP's own IT organization has also adopted the client/server model and has determined that it will be the basis for our company-wide information architecture.

Appendix

The Application Processing Split

Earlier in this document client/server computing was described as a the distribution of a single task or or application between two or more systems. This is a simple, but accurate definition. Some consultants will describe the distribution of the processing task in more detail. Some of your customers may have attended seminars hosted by these consultants or read articles where are more specific model of client/server computing is discussed. This section is intended to make you familiar with the different architectures that your customer may be aware of and may refer to as client/server.

Client/Server Implementations


Data Management
Data Management
Data Management
Data Management
Data Management
Application Logic
Application Logic
Application Logic
Application Logic
Application Logic
Presentation
Presentation
Presentation
Presentation
Presentation

Source: Gartner Group

The diagram above depicts five logical division points where an application process can be split for processing on a client and a server. These division points are based on a model where a given application process has three distinct functions: presentation, application logic and data.

Three of the client/server configurations result from a split **within** a particular function (i.e. a split within the presentation function is seen in a host/terminal type set up where a GUI resides on both the host and the desktop device making the host application more user-friendly; a split within the data management function is exemplified by distributed databases which are used to optimize end-user access to the information they need.) Two additional configurations result from a split between the application functions, having the full presentation function residing the on client such as occurs when an X-terminal is used as a client, or having the complete data management function reside on the server resulting in its sole function as a data repository.

Purists will say that "true" client/server computing is the case where the application logic is split between the two processors. For example: An application running on a client requests specific information from a server, perhaps all records with a balance of more that $10,000 from a certain customer file. If the application processing split was between the application logic function and data management functions the server would return the entire customer file for further processing by the client. In a "true" client/server approach, where the application logic itself was split, the server would search for and return only those records which fit the request of the client reducing the network traffic and decreasing response time.

Most applications available today that claim to be client/server applications would not fit the purists' definition of client/server. As seen in the applications list in this document, many vendors have added a GUI to there existing centralized application and call this a client/server application and according to the simple definition, it is. The oil company example in this document describes how HP offered one of our customers client/server computing advantages by adding a GUI to their mainframe applications. Other customers like the telecommunications company, have added capabilities for PC- or workstation-based applications to utilize data that resides elsewhere.

The examples illustrate evolutionary steps toward a purists' view of client/server and offer significant benefits to customers who are used to text-based centralized applications and customers who are using PC's who previously have had little access to data on other systems. One of the advantages of HP's client/server implementation is that we can build a client/server environment based on a customers existing technology infrastructure. This evolutionary approach means that customers can evolve to a true client/server model at their own pace and as their business situations dictate. Be sure to evaluate your customer's existing technology and application processing environment before suggesting a particular client/server implementation.

Paper Number 2007

Why did my Backup Fail ?

Wolfgang Friedrich

HEWLETT PACKARD
Network & System Management Division
Herrenberger Str. 130
D-71034 Böblingen
Germany

Tel. +49-7031-143754

Introduction

Most of the operators responsible for backup have probably seen messages like:

***Error:** Cannot access device*

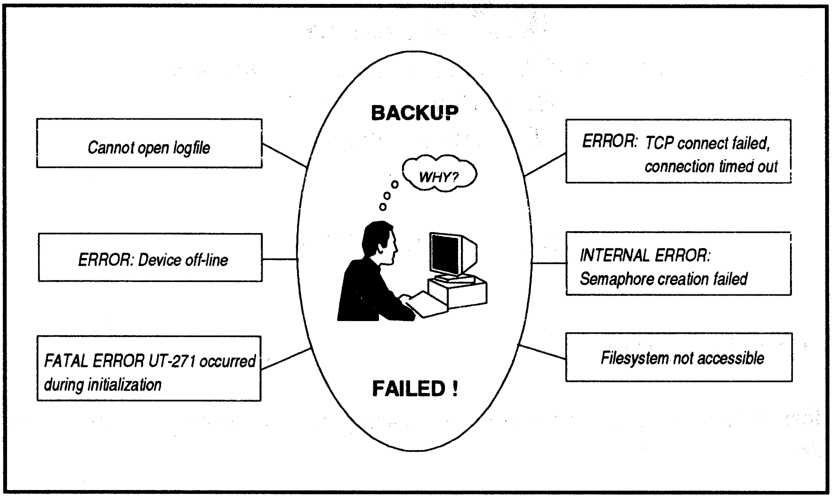
***Warning:** Some of your files have not been backed up*

***Fatal Error SE-2314** occurred in proc.media.*

Some of these messages might be self-explanatory and easy to handle. However, experience shows that most of these error conditions can lead to a troublesome situation. Especially in unusual error situations the result can be hours and hours of troubleshooting.

Because of the complexity of today's backup solutions there are lots of potential problem areas. This paper will identify and discuss the most important areas that could lead to a failure of parts of your backup or in the worst case of your complete backup. These areas include problems with your backup product, the underlying network, device problems, restrictions of computer resources and the whole area of backup policies and their use.

In addition to potential problem areas during backup, the paper will also discuss problems which can occur during restore operations. Recommendations will be made to guarantee trouble-free backup and restore operations.


BACKUP
WHY?
FAILED !
Cannot open logfile
ERROR: Device off-line
FATAL ERROR UT-271 occurred during initialization
ERROR: TCP connect failed, connection timed out
INTERNAL ERROR: Semaphore creation failed
Filesystem not accessible

Backup and recovery in complex environments

In today's system environments, backup is usually a task that is performed in networks of different machines. These machines may include entry-level systems like PC's, mid-range systems like workstations and high-end systems like mainframes. Several heterogeneous systems are connected on a local or wide area network. A typical UNIX environment may contain a huge amount of data, even data that must be on-line for 24 hours a day. Networks are growing every day both in terms of their capacity and in the type of machines and devices they support.

The backup in such an environment may be a network file system backup of multiple machines. In this case backup devices are shared. Another approach is a local high-speed backup of raw data where no networking is involved. No matter which backup method or backup tool is being used, the major requirements for the backup operation are:

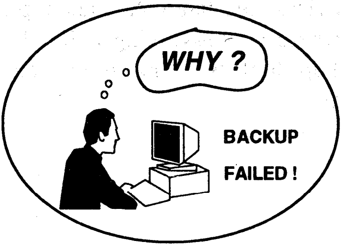
- Data Integrity and Security
- Reliability
- High Performance
- Implementation of Full and Incremental Backups
- Some kind of Media Management
- Complete Backup and Restore Management System

Most system managers expect a complete backup in a network environment to be managed by their operators. However, because of the complexity of the underlying hard- and software, errors and instabilities may occur in different areas. I want to distinguish between the following potential problem areas:

- The Backup Product
- Computer Resources
- Network Services
- The Backup Policy
- Backup Media
- Backup Devices
- Restore Operations

The Backup Product

The operating staff must have a thorough understanding of the backup product. Especially for restore operations, which occur less frequently than backup operations, the operator needs to understand the process. It is very easy to setup a restore command that accidentally overwrites a complete directory structure or a complete disk. In a data center environment, the operators need to be well trained on the product and understand the underlying concepts. The software must be correctly installed.


WHY ?
BACKUP
FAILED !

The Backup Product & Computer Resources

- **Backup software not correctly installed**
- **Backup software does not work with OS release**
- **System parameters like shared memory not sufficient**
- **Not enough disk space for log entries**

A common source of errors with the backup product itself is the whole area of software updates. These updates can involve the backup product, the operating system or other software necessary to execute the backups correctly. Especially for network backups, you must ensure that all layers of network software necessary for the backup are up and running. When performing an operating system update, you must check that your backup product supports this new release and that it is compatible with older operating system versions in the network.

The backup product must be able to handle files that are larger than one medium. For example if you have a file that is 400 MB in size, then the backup product must be able to span that file over multiple MO (Magneto Optical) disks which hold only 325 MB per disk.

Computer Resources

Some backup products use computer resources extensively. For example, if you use a product for a high-speed local backup to multiple backup devices, you can achieve a backup performance of up to 20 GB/h. In order to achieve this performance, some computer resources will exceed the default values. These resources can be parameters for shared memory or semaphores.

If during the backup, log entries are created for all files that are backed up, you must make sure that enough disk space is available for these log entries. It will be very annoying if your backup fails just because you experienced a filesystem or database overflow due to the log entries.

Another area of trouble might be unmounted filesystems. If a disk is unmounted and that disk holds a file system, this file system will not be backed up if the backup is a filesystem backup. In order to avoid that situation you must make sure that all disks listed in */etc/checklist* are mounted on your system. During a restore, it might be even worse if you try to restore a filesystem that is currently not mounted. For example, if your */usr* directory is on a separate disk and it is unmounted during a restore operation then the restore will be done to the root disk. This could lead to a disk overflow problem if you are not aware of this situation. To guarantee a trouble-free restore you should again check */etc/checklist* to find out whether all disks are mounted.

Network Services

Before you do a backup in the network, you must make sure that the appropriate network software is up and running in your environment. For the initial setup of your backup environment, you must configure several machines in your network that you want to backup from a central machine. The network services on these machines must be configured accordingly and the central backup machine needs access to the other machines in the backup environment. These machines can be machines that are being backed up or machines to which a backup device is attached.

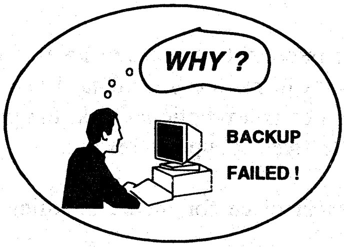
In environments with distributed filesystems, you can also do your backup with NFS. However, some older versions of NFS may cause problems when restoring files owned by *root.* The preferred way of doing network backups is always to use a backup product that launches a process locally. This process is responsible for reading the data to be backed up and sending it over the network to the machine with the actual backup device.

In heterogeneous networks you must ensure that the appropriate services are configured for the backup operation. It is strongly recommended that you use a backup product which really exploits the features of network services. These network services could be Remote Procedure Calls (RPCs) or data interchange on the TCP/IP level. If your backup relies on tools like *cpio* or *tar,* you will soon reach limits in terms of network support and other features necessary for backups. These tools have been developed for data interchange in the pre-network era, e.g. for transfer of data from a machine of vendor A to a machine of vendor B. Some people use complex scripts around these tools to expand their usage in a network environment. But since these tools were neither designed for network usage nor for the backup requirements of today the result is always more like a workaround than a sophisticated backup solution.

The Backup Policy

In workgroup environments there can be several people responsible for backup policies and procedures. However it is helpful if there are overall guidelines for the backup procedure. In this case a company could internally train their personnel on this procedure. With this approach it is easier to share resources, people and computer machinery.

One of the major tasks in designing a backup policy is to determine which data needs to be backed up. Typically the important data is user data and application data. The operating system itself does not need to be backed up since it is recoverable from install tapes. When determining what files need to be included in the backup procedure you must be careful to avoid a possible data loss in the event of corrupted data on the hard disk. Some backup tools allow end users to specify data to be backed up. In this case it is a question of educating the end users to understand what they are requesting for backup. It is the end user's responsibility to make sure that all important user and application data is included in a secure backup. Duplicate data that is being backed up from a different place should not be included in the backup.


WHY ?
BACKUP
FAILED !

The Backup Policy

- **Not all mission-critical user data included in backup**
- **Important configuration files like /etc/rc not backed up**
- **Backup frequency not appropriate**
- **End users excluded important data from the backup**

The other important issue of a backup policy is the backup frequency. Determine how much data you can afford to lose and plan your backup frequency accordingly. You should also decide if it is really necessary to backup your data every night. If your data changes only infrequently you might choose a backup only every other night.

If your environment requires a minimal time window for complete filesystem restores, you must adjust your schedule of full and incremental backups accordingly. For example, if you do full backups every Monday and incremental backups every Tuesday through Friday, it could take hours to restore a complete disk. There are alternatives like more frequent full backups or local high-speed backups.

Errors in the setup of the backup policy can have very bad consequences. For example, if a user's directory is not or only partially included in the backup, the data can be lost in the case of a disk failure. Even if you don't backup the operating system, you must include specific files, like */etc/passwd* or */etc/rc*. These files are important for the overall system configuration. It will be very time-consuming to recover a system if the */etc/passwd* file gets lost.

Backup Media

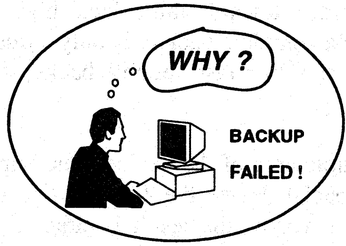
Concerning backup media an important issue is that you are able to easily identify the medium from which you want to restore your data. Even if the restore operation is sometimes considered as an exceptional task, the process of media handling is essential for a trouble-free backup and restore.

There should be a well defined process in place for media handling. This process should cover issues like:

- Labelling
- Initialization of media
- Physical storage
- Deletion of obsolete media

The labelling of the media must be correct and consistent. The media should have physical label stickers as well as virtual label information at the beginning of the tape. There must be a clear and simple way to identify a particular medium. The label should consist of information based on the current backup date and the data contained on this medium. In order to guarantee that media is not used for another purpose the media should be stored in a safe place physically separate from the actual machine environment. If you use a medium for the first time you should make sure that the medium is initialized by the backup tool before the actual backup. In this way you make sure that the medium can be easily identified as a medium for backup purposes.

There must also be a process in place to discard backup media that will not be used for backup in the future. Otherwise there is a chance that you could load the tape for a restore and not find the data that you actually want to restore. The media must guarantee good data integrity and good shelf life. There should be a procedure to determine how often a specific medium has been used. Depending on the kind of medium, it should not be overwritten again after it has passed a certain age or usage threshold.


WHY ?
BACKUP
FAILED !

Backup Media & Backup Devices

- **Incorrect labelling**
- **Capacity of media insufficient**
- **Device file not setup or not specified correctly**
- **Device driver not present**

One possible cause of a backup failure is that the capacity of your backup medium is insufficient. You will then see a message telling you that only part of the scheduled data has been backed up. In terms of capacity per medium, the DDS technology offers a cheap and reliable solution. If, however, you want to perform unattended backup of large amounts of on-line data, you might choose some kind of repository system. In this case the MO (Magneto-Optical) autochanger might be an alternative, since these devices offer an on-line capacity of up to 100 GB.

Backup Devices

For most environments the support of a variety of backup devices is available. These devices include the more traditional 1/2 inch 9-track tape and 1/4 inch 16- or 32-track tape drives as well as newer technology like the DAT (DDS Format) tape drives and the MO (Magneto Optical) disk drives.

A new compression DAT drive by HP offers a capacity of 8 GB. This amount of data can be stored on a 90 meter DDS tape in a little more than 1 hour. With this fast developing technology the trend is towards more local high-speed backup solutions. In some environments a complete backup is only a question of minutes. The advantage is that users can do a fast, reliable backup that is cheap and easy to handle.

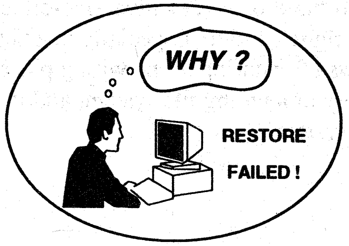
Another area of trouble might be the whole area of the device files. Suppose, you do a backup to a drive with the special device file */dev/rmt/0m*. If this device file has been erased for whatever reason, the result is either a failed backup or a backup into a regular disk file called */dev/rmt/0m*. This can cause a filesystem overflow. The same behavior occurs when the device file name is misspelled, e.g. */dev/rmt/om* instead of */dev/rmt/0m*.

Another common source of errors is that the backup user does not have write access to the device file or that the device driver for the backup device is not present. In this case the backup will fail immediately.

Restore Operations

A lot of issues concerning restore operations have already been discussed. However, I want to emphasize this particular area, since several unexpected errors can occur during such an operation. I am sure that most operators have already been in a situation where they couldn't do a requested restore or experienced other unexpected side effects.

Before performing an actual restore operation there must be a well defined way to identify the medium that holds the data that needs to be restored. Usually, this is done through the logging information made during the backup. This logging information might be available on-line on a computer system or off-line on paper or on any other medium. No matter where the information resides, there must be an easy and fast way to access it.


WHY ?
RESTORE
FAILED !

Restore Operations

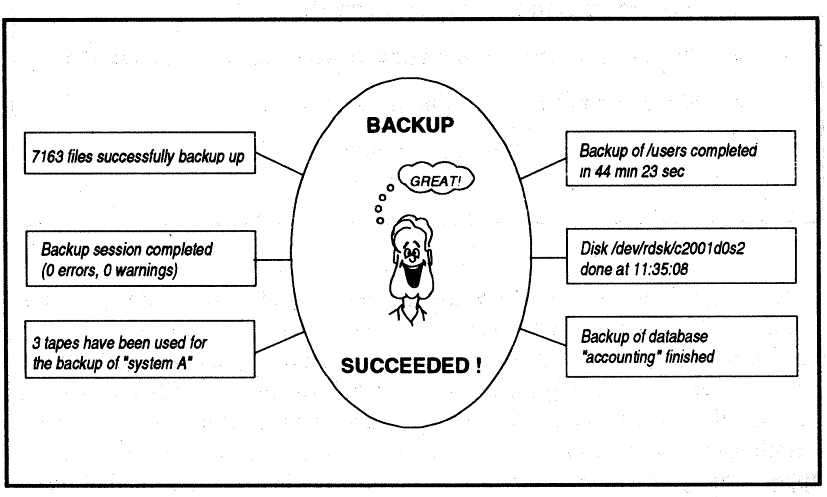
- **Restore deleted the directory structure**
- **File system overflow because deleted files are not tracked**
- **File could not be identified on tape**
- **Restore done to the wrong machine**

A very sensitive area during the restore process is conflict resolution. A restore conflict exists if the files or directories found on the backup medium already exist on the computer's disk. If the operator did not explicitly specify that he wants to overwrite existing files and directories, the restore will simply not be done. On the other hand, if the operator specifies overwriting during a restore operation, it is very easy to destroy a large amount of data on the disk. Just suppose that you restore a file */users* to a disk with a directory called */users*. During the restore the whole directory */users* with all underlying directories and files will be removed and replaced by the file */users*.

Another problem can be unexpected file system overflows. Some reasons for that I have already discussed in the section about computer resources. An additional problem can show up when the backup product does not keep track of deleted files. If all data from a full backup tape and the subsequent incremental backup tapes is restored without the operator taking note of the files that have actually been deleted from the computer's disk, a file system overflow might occur.

When the restore command is entered, the operator must be careful to restore only the precise data necessary. Otherwise, restoring less data will not satisfy the users, and restoring more data could have other negative side-effects. In addition, the data must be restored to the right location. Especially in a network backup environment, there are many ways of restoring to the wrong place. The data could be restored to a wrong directory or a wrong file system, and it might even be restored to the wrong computer system!

Summary



BACKUP
GREAT!
SUCCEEDED !
7163 files successfully backup up
Backup session completed (0 errors, 0 warnings)
3 tapes have been used for the backup of "system A"
Backup of /users completed in 44 min 23 sec
Disk /dev/rdsk/c2001d0s2 done at 11:35:08
Backup of database "accounting" finished

This paper has provided an overview of potential problem areas during backup and restore operations. It is the responsibility of system administrators and backup operators to guarantee a hassle-free operation. In some environments even the end users might be responsible for a successful backup. Most of the issues can be solved through careful planning of policies and procedures. Since backup is one of the most sensitive areas in data center management, the establishment and monitoring of the defined policies and procedures is essential.

PAPER NUMBER: 2008

TITLE: AWK Programming

PRESENTER: David Totsch
La Salle National Bank
181 West Madison
Chicago, IL 60602

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

High Speed Network Transition

Mohammad Malik
Worldwide Customer Support Operations
Information Technology
Hewlett-Packard Company
100 Mayfield Avenue
Mountain View, CA 94043 USA

Sam Sudarsanam
Professional Services Division
Hewlett-Packard Company
100 Mayfield Avenue
Mountain View, CA 94043 USA

Introduction

A complex, evolving global economy is creating interdependencies among different countries and organizations. With this comes the need for timely, accurate transfer of high quality multimedia information -- implying information age capabilities for everyone. Application users will require information access and communication capabilities in any combination of media -- voice, data, image -- anytime, anywhere, in a timely, cost-effective manner. High-speed networks and distributed systems are critical needs in this area of enterprise integration.

Limitations of Current Networks

Over the past several years, the role of Local Area Networks (LANs) has changed considerably. For example, many corporations are organizing people into workgroups by project or business unit rather than geographical location. These workgroups require control over their own resources as well as access to corporate resources on the network. The growth in numbers and speeds of the attached computing devices is taxing LANs to the point that networks are beginning to run out of bandwidth. The problem is expected to become more acute as distributed computing and multimedia applications are implemented.

Today's architecture, built on hubs, bridges, and routers incorporating distributed or collapsed backbone concepts, have not been able to provide the needed flexibility. The limitation of today's LANs has introduced performance and congestion problems that become more pronounced by shared media access. The popular solutions of segmenting and resegmenting LANs have not completely resolved the bandwidth issues.

A long term innovative architectural solution is needed to address the LAN bandwidth problem. A lot of work is being done in the development of Asynchronous Transfer Mode (ATM) and Fiber Channel switching fabrics to address these issues.

Current Generic LAN Internet Topology

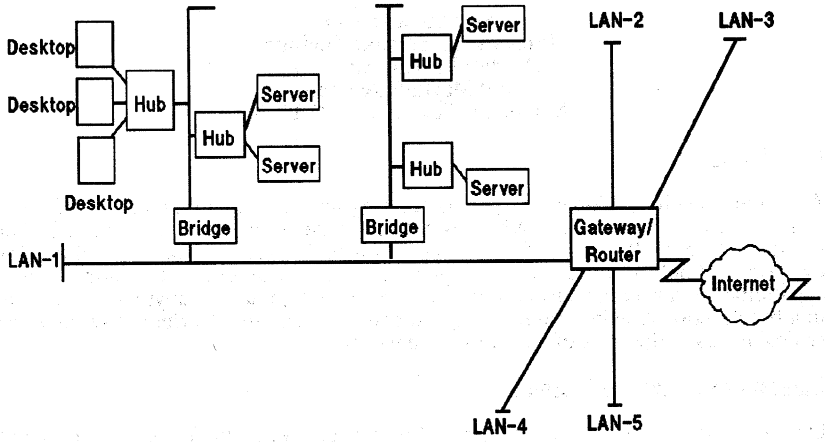


Figure 1. An existing LAN topology based on shared media hubs, bridges, and routers.

The real challenge facing the network managers today is to develop a cost-effective migration path that will fulfill the short-term requirements and lead them toward building the ATM infrastructure. The following sections will discuss the switching hub and ATM technologies as well as briefly describe Fibre Channel and Photonics.

Switching Hubs

Conceptually, switching hubs represent the first step towards migration from shared media access to switched LANs. A switching hub is more like a multiport bridge rather than a multiport repeater, with routing protocol support. It uses an intelligent filtering mechanism to forward traffic to only those ports that need to hear it . This is essentially done at the media speed, that is, they are just as fast as the cabling media to which they are connected. This combination of multiport switching (filtering) and media speed moves us from shared to switched LANs. In a shared LAN, all ports on the hub compete for the same 10 Megabits per second (Mbps) bandwidth. In a switched LAN, each port or a combination of ports on the hub may be configured to see the equivalent of a dedicated 10 Mbps.

The switching hubs extend the life of existing 10 Mbps Ethernet networks. They will provide network managers with enough breathing space until 100+ Mbps technologies (100 Mbps Ethernet, 100BaseVG, ATM LAN and Fibre Channel LAN) become available. These hubs also provide the flexibility of network reconfiguration. For example, critical servers could be assigned dedicated per port bandwidth. Workgroup networks may be configured for reduced congestion or broadcast problems. Diskless clients and X-Terminals are sensitive to network performance. Switching hubs provide options to avoid these problems. They can be configured to be less susceptible to network performance degradation due to variables such as delay and traffic levels. As a part of their technology migration strategy, switching hub vendors are developing ATM backplanes for adding cell-switching fabric and multimedia support.

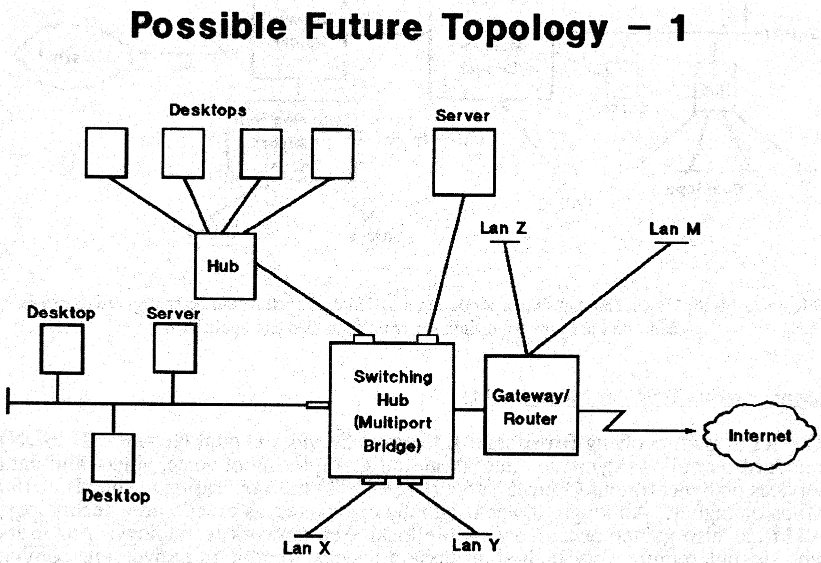

Possible Future Topology - 1
Desktops
Server
Hub
Lan Z
Lan M
Desktop
Server
Switching Hub (Multiport Bridge)
Gateway/ Router
Internet
Desktop
Lan X
Lan Y

Figure 2. As numbers and speeds of attached LAN devices increase, switching hubs offer an interim solution to bandwidth bottlenecks.

Possible Future Topology – 2

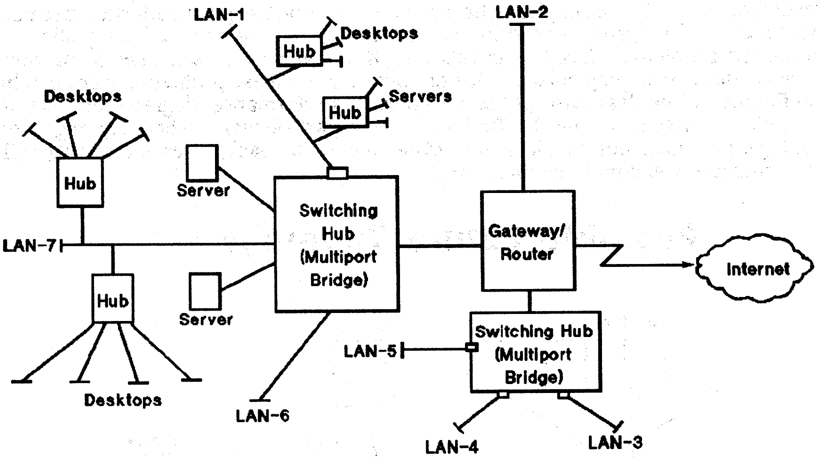

LAN-1
Hub
Desktops
LAN-2
Desktops
Hub
Servers
Hub
Server
Switching Hub (Multiport Bridge)
Gateway/ Router
LAN-7
Internet
Hub
Server
Switching Hub (Multiport Bridge)
LAN-5
Desktops
LAN-6
LAN-4
LAN-3

Figure 3. Multiple switching hubs incorporated in a LAN infrastructure can be configured to provide dedicated per port bandwidth for critical systems and applications.

Asynchronous Transfer Mode (ATM)

The ATM is an evolving Broadband Integrated Services Digital Network (B-ISDN) standard that allows dynamic integration and multiplexing of voice, video, and data services on Synchronous Optical Network (SONET) links operating at speeds of 150 Mbps or higher. Although, it was originally envisioned as a wide area technology, ATM has also gained acceptance as the local area network technology. For users who do not require very high transmission speeds, vendor initiatives are coming from several directions to make ATM products and services available at lower speeds. Several product vendors have presented the ATM Forum with proposals for low-speed ATM for local area networks. For example, AT&T and Hewlett-Packard Company have jointly presented a 51 Mbps proposal.

ATM allows dynamic bandwidth allocation on demand. It uses fixed length cells and is referred to as cell relay. The fixed length, 53-byte ATM cell consists of 5-byte header and 48-byte information field (payload) (Figure 4). ATM network routes traffic by encapsulating 48-byte information payload of fixed size cells and switching these cells through very fast packet switches. It supports data rates of 150 Mbps, 600 Mbps, and potentially several Gigabits per second (Gbps) speeds.

ATM Cell

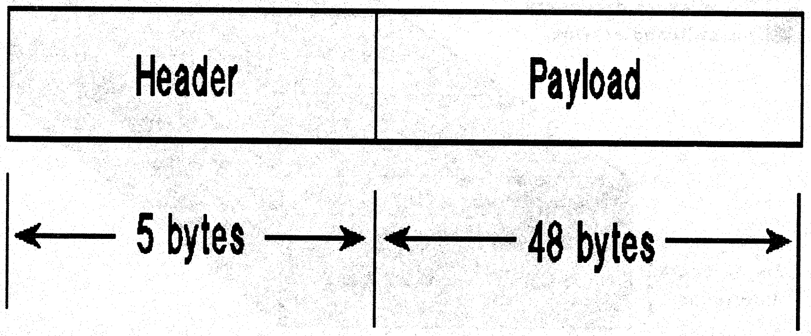


Figure 4. The fixed-length, 53-byte ATM cell consists of 5-byte header and 48-byte information field (payload).

As illustrated in Figure 5, ATM protocol operates over a physical layer. The protocols immediately above the ATM layer are used to adopt various services. The control plane with its layered structure provides signalling and connection control functions. The user plane transfers application information, whereas the management plane provides maintenance, operation, and other management functions.

ATM Layers

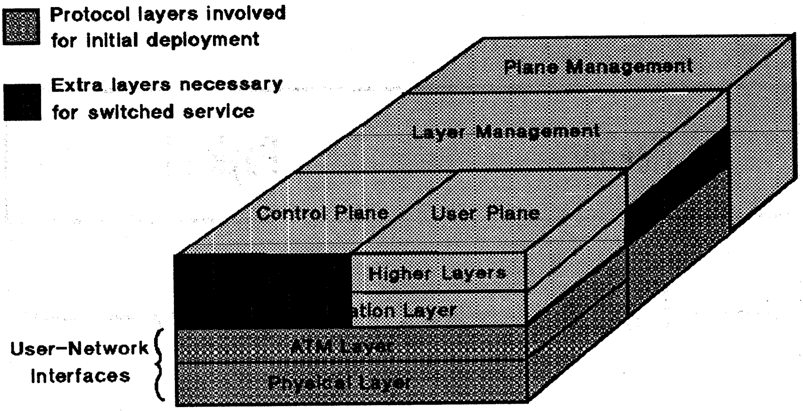

▨ Protocol layers involved for initial deployment
■ Extra layers necessary for switched service
Plane Management
Layer Management
Control Plane
User Plane
Higher Layers
tion Layer
ATM Layer
Physical Layer
User-Network Interfaces

Figure 5. ATM layered architecture model.

How does ATM Fit in Existing Networks?

In the long run, the eventual vision calls for a homogeneous end-to-end, cell-based infrastructure (Figure 8). However, cell switching would initially be implemented in the backbone and high-performance workgroups without completely replacing existing networks. Current protocols and applications as well as wiring plants can remain unchanged. ATM switches can be selectively deployed where they solve acute problems, thereby protecting investments in existing network components like shared bandwidth hubs, routers, bridges, and switching hubs. Heavily utilized resources such as high-end application servers can be provided direct ATM connections.

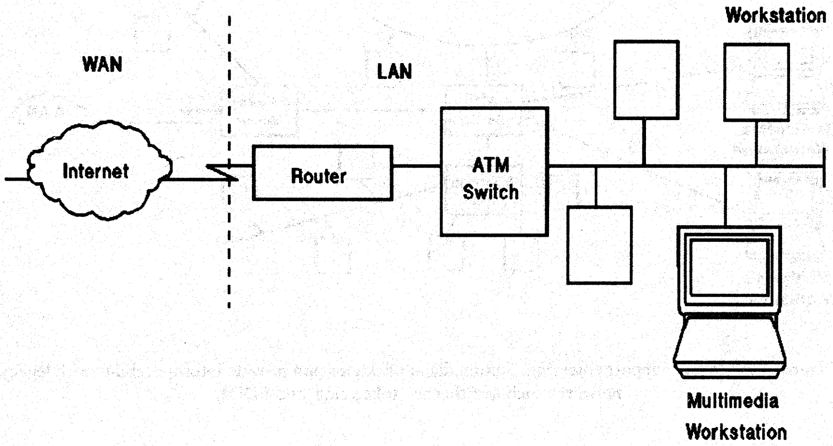
In short, the local ATM can be deployed for the following reasons:

1. Backbone ATM for connecting hubs, routers, and bridges.

2. Workgroup ATM for distributed client server computing deploying high-end servers and workstations.

3. Wide area ATM network connectivity.

One of the more important attractions of ATM technology is its scalability and growth path. It can utilize any combination of speeds, from DS-3 (45 Mbps) to SONET OC-48 (2.5 Gbps), over a variety of media such as twisted pair and fiber optics.


Generic ATM LAN – Example 1

WAN
LAN
Internet
Router
ATM Switch
Workstation
Multimedia Workstation

Figure 6. ATM switch can be integrated into existing infrastructure to support critical LAN segments.

Generic ATM LAN – Example 2

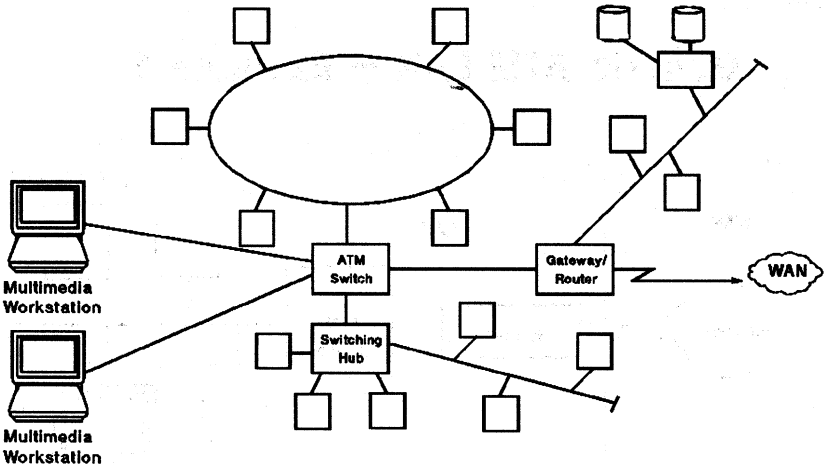

ATM Switch
Gateway/ Router
WAN
Switching Hub
Multimedia Workstation
Multimedia Workstation

Figure 7. ATM will support emerging multimedia applications and provide interoperability with legacy networks such as Ethernet, token ring, and FDDI.

Generic ATM LAN – Example 3

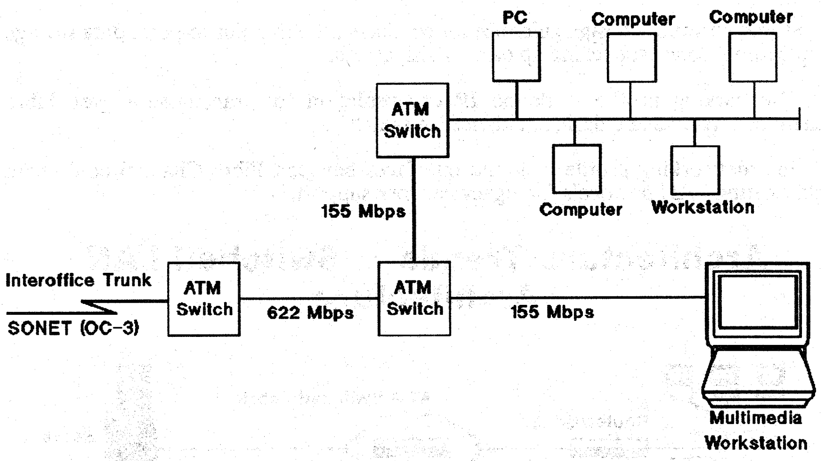


Figure 8. Eventual vision of an end-to-end ATM and SONET network that would also integrate legacy networks and systems.

Fibre Channel

Fibre Channel standard is being defined by the American National Standards Institute (ANSI). Some of the industry experts believe that the channel-attached architecture that is usually associated with host-peripheral connections could emerge as a viable alternative to today's high-end LAN architecture. Recently it has enlisted support among computer manufacturers like Hewlett-Packard Company, IBM, and Sun Microsystems. Fibre channel provides three types of connections: point-to-point I/O (example: mass data storage); clusters of high-end workstations; and high-speed switched LAN connections.

Fibre channel product vendors believe that the high-speed channel architecture operating at up to 1 Gbps can be converted by adding a switch specified in the new standard. Channel-based adapters do most of the processing in hardware and hence, achieve high speeds and low delays.

Fibre channel specifications define an encapsulation technique to handle both LAN and high-performance channel protocols. Fibre channel supported interfaces include High Performance Parallel Interface (HIPPI), Small Computer Systems Interface (SCSI), and Intelligent Peripheral Interface (IPI). Also, Fiber Distributed Data Interface (FDDI), ATM, Token Ring, and Ethernet LAN traffic will be supported. In order to gain acceptance as a network solution, Fibre Channel will need to ensure interoperability among different vendors' products. Recently, a

Fibre Channel Systems Initiative (FCSI) has been formed by Hewlett-Packard Company, IBM, and Sun Microsystems to address the interoperability issue. The FCSI is developing three profiles providing enough details for vendors to produce interoperable products. These profiles include:

1. Storage profile -- design guidance for products used in point-to-point data storage applications such as, backing up data to disk arrays.

2. Networking profile -- define IP encapsulation for transmission over Fibre Channel networks and design guidance for switches.

3. Internetworking profile -- define interfaces between Fibre Channel equipment and existing LAN protocols for legacy network support.

Architecture Trends – Switched LAN Architecture

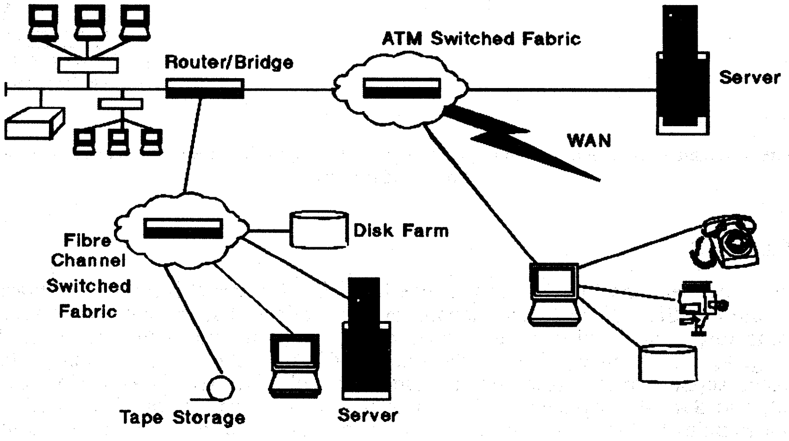


Figure 9. An integrated hybrid network of Fibre Channel and ATM switching fabrics interconnected via bridge/router. ATM provides multimedia communications support and LAN to WAN integration. Fibre Channel provides mass storage integration and high-end workgroup cluster support.

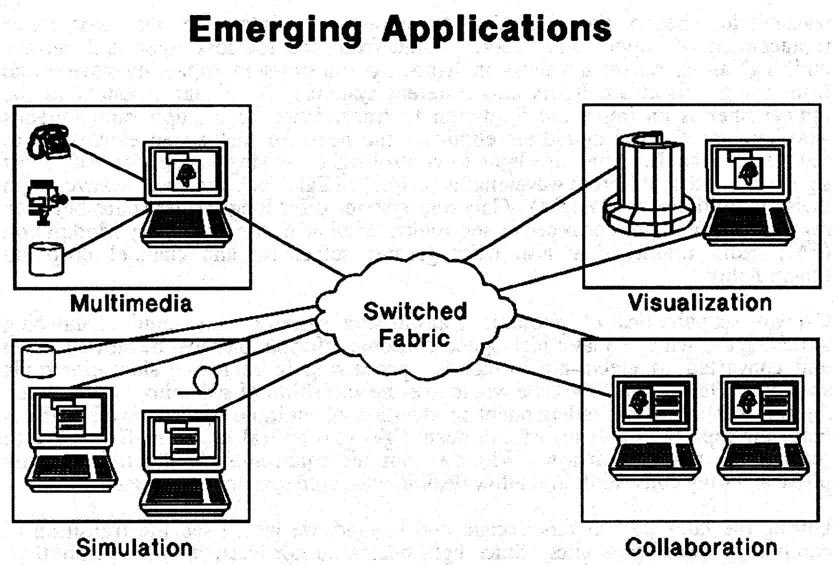


Figure 10. Switched Fabrics that will provide enterprise-wide integration of high-performance systems. This would enable efficient implementation of emerging applications such as client/server, imaging, collaboration, simulation, and multimedia.

Photonics

Photonic computing, switching, and transmission are the three technology drivers that will have tremendous impact on high performance interconnection networks of the 1990s and beyond. Next generation of computing would require development of systems capable of Teraflop processing rates, Terabit per second (Tbps) communications speeds and Terabyte storage capacity. Effective architectures will balance resources between communications, processing, and storage system components. One of the important considerations in successful high-performance systems is the scalability of the architecture, potentially extending over global scales for multiprocessor applications. This will place increasing demand on the interconnection network. All-electronic interconnects impose a communications bottleneck, since the link bandwidth and the physical distance these links can cover are limited by power dissipation and electronic crosstalk. Optoelectronics and photonic technologies could play a critical role in the future architectures of the highly scalable and flexible interconnects for multicomputer parallel processing systems.

Distributed computing, multimedia teleconferencing, remote interactive education programs, high-definition TV, and two-way switched video on demand are some of the applications that would provide market place incentives for the evolution of an all-fiber private and public network including fiber to the home. During the 1980s, transmission began the transition to photonics, leading to the progressive replacement of copper with fiber. Aside from the low-loss fiber and surface emitting lasers, recent advances in lightwave transmission capability have come from using optical amplifiers and coherent systems. Nonlinear crosstalk in the optical fiber is an important limitation to transmission of multiple simultaneous wavelengths. Optical amplifiers eliminate the need for high-speed electronics to regenerate signals. They use light to control light. A single amplifier will boost signals carried by different wavelengths (colors) of light, as is the case in wavelength division multiplexing (WDM). Coherent systems offer longer fiber spans between regenerators and greater receiver sensitivity. Similar to the Frequency Modulation (FM) radio tuning, they also offer greater selectivity and channel drop-add functionality.

Growing capabilities of photonic transmission systems also require matching switching capacity. Today's high-speed photonic information must be slowed down and converted to electronic format for processing in relatively slow electronic switches. During the 1990s, we would see the transition of switching to photonics. Device technology for independent modulation of multiple optical wavelengths is maturing rapidly. Laser arrays can beam light into optical media or free space to reach other photonic arrays. Wireless photonic connections would free us from physical wiring constraints and allow flexible and rapid systems reconfiguration.

During the later part of this decade and beyond, we would see the transition of computing toward photonics. Since light beams do not interact with one another, massively parallel computing architectures should be possible. Wireless photonic interconnections will provide nearly instantaneous computing. We are at the threshold of developing Integrated Photonic Circuits (IPCs) that may rival Integrated Electronic Circuits (IECs). Experimental digital photonic processors offer the promise of computers with orders of magnitude higher processing power of today's electronic machines.

Concluding Remarks

Switching hubs are becoming part of the LAN infrastructure for high-end computing and bandwidth intensive applications. This will allow users the option of squeezing one to three additional years out of investment in existing LAN technologies. Eventually, these hubs will incorporate ATM backplanes. ATM LAN switches and ATM switching hubs will combine to deliver high bandwidth on demand to the desktop. End-to-end ATM is a vision with different perspectives. Some industry experts believe that ATM will start from LANs and move out into the WANs; whereas, others are of the opinion that ATM will move from WAN to LAN. Despite these differences in outlook, one can visualize an integrated systems network that will link legacy systems and network components (routers, bridges, and hubs) with ATM and Fibre Channel switching fabrics.

Historically, networks have migrated from Kilobits per second (Kbps) (host to terminal connection) to Mbps LANs (Ethernet, Token Ring, and FDDI). Currently the Gbps network infrastructure is being defined in the form of ATM LAN/WAN and Fibre Channel standards. Optoelectronic and photonic technologies would potentially provide Tbps communication speeds. The future paradigm shift brought about by photonic computing, switching, and transmission would move us closer to offering universal information services -- the ability to provide any combination of voice, video, and data anywhere, anytime with convenience and economy.

Acknowledgements

The authors wish to convey special thanks to Ann Dingwall, Manager, PSD Customer Education Engineering and Tim Farley, Manager, UNIX/Network Engineering, WCSO IT, for their valuable technical feedback. Also, special thanks to King Wah Moberg, Learning Products Engineer, PSD Customer Education Engineering for her editorial assistance.

References

1. Biagioni, Edoardo et al., "Designing a practical ATM LAN," IEEE Network Magazine, March 1993.

2. Costa, Bruno., "Lightwave Communications: A Still Evolving Technology," IEEE Communications Magazine, June 1993.

3. Dagres, Todd., "Switching to a LAN-in-a-Box," Business Communication Review, January 1993.

4. Estes, Glenn., "ATM Networking What's Been Agreed?," Business Communications Review Supplement, February 1993.

5. Heilmeir, George., "Global Begins at Home," IEEE Communications Magazine, October, 1992.

6. Herman, James et al., "ATM Switches and Hubs Leads the Way to a New Era of Switched Internetworks," Data Communications, March 1993.

7. Mayo, John., "The Telecommunications Revolution of the 1990s," Scientific American, Science in the 20th Century, Special Issue, 1991.

8. Sauer, Job et al., "Photonic Interconnects for Gigabit Multicomputer Communications," IEEE LTS Magazine, August 1992.

9. Saunders, Stephen., "Fibre Channel Outraces ATM," Data Communications, May 1993.

Paper # 2010

Measuring and Monitoring Business Transactions in an Open Systems Environment

Wayne Morris
Hewlett-Packard Company
Software Technology Division
Roseville, California
(916)785-8212

Abstract

To effectively manage a computing environment we need to understand and monitor workloads, response times and throughputs from the business perspective. Distributed, heterogeneous computing environments present additional challenges to achieving this objective. In a distributed, client-server application, a single business transaction may require several interactions between the user and the desktop as well as requests from the desktop to one or more servers. We therefore need to identify, measure and monitor business transactions and user-perceived response times by application.

This paper examines the need to measure and monitor business transactions and explores current and future technology to enable this in distributed open system environments.

Background

Open system environments incorporate system and network facilities which communicate and interoperate to support distributed applications. These facilities will probably be heterogeneous in nature and supplied by multiple vendors. Distributed applications (including client-server applications) are developed as cooperating modules which reside on different systems within the network, and work together to provide the functionality users require to achieve their business objectives.

More corporations are implementing distributed processing built on open systems and utilizing client-server applications. These distributed applications support all aspects of the business including technical, commercial and manufacturing activities, and are increasingly critical to the success of the business. In these environments a greater emphasis is being placed on the ability to effectively manage distributed applications. Monitoring and optimizing the performance of these distributed applications is a very important element of this management. The objectives of performance management are to accomplish the following:

- ensure user productivity is not negatively impacted by the service levels (availability, response times, job turnaround times) provided by the computer facilities;
- optimize the return on investment in the computer facilities;
- plan for support of future business goals and strategies by the computer facilities.

The remainder of this paper will focus on each of these objectives within open system environments.

Measuring User Productivity

User productivity is usually measured in terms of the user's ability to complete an amount of work within a given time period. Another important factor which should be considered when measuring user productivity is responsiveness to customer requests. Adequate response to user requests provides a competitive advantage particularly in industries where direct customer interaction is required to complete a transaction, and where customer service is an important differentiator between competitors. Examples are the banking, airline and car rental industries.

In computing terms the first measure above can be thought of as *throughput*, while the second is related to *response times* and *availability*.

Measuring Computer Service Levels

Traditionally, the service levels provided by computing facilities are measured from a systems perspective. The typical measures of system responsiveness are response time measures such as first response and response to prompt. These take a very simplistic approach using a single system view. First response is how long the user waits until something appears on the screen following a request for service, while response to prompt is how long the user has to wait before another command can be issued following a request for service.

These measurements are captured by operating system instrumentation which can identify events such as terminal read and write requests, and terminal read and write completions. The time intervals between these events is used to derive response time values. This method can suffer problems on a single system where there is not a consistent definition of transactions which trigger these events, or where more than one system is involved in supporting distributed applications. Response times captured in the above fashion on UNIX-based systems for example can represent single characters being echoed, or blocks of data being transferred between the user terminal and the system. In the case of a distributed application, response times as reported on a single system may have no meaning or value. Consider the case of a client-server transaction where the user-interface presentation and a local database is supported on the user's local workstation, while the main transaction processing engine and corporate database is supported on another system elsewhere in the corporate network. In this case having a measure of the response from the local system is meaningless unless we can also capture and correlate the response from the server system.

The other difficulty in measuring response time is that typically we would like to know response times by transaction by application, whereas most operating systems provide response times either on a global basis (aggregated for all applications on the system) or for an individual process. This again can limit the usefulness of the response time data as there may be a number of processes that make up a single application, and we may want to see response times aggregated by user application. This is important where consistent response time is required for a number of transactions within a single application, and we need to easily compare response times across transactions for each application.

The above problems can, to varying degrees, be overcome by post processing the captured performance information, however this makes it difficult to effectively measure and monitor application performance in an on-line real time fashion. The post processing approach by itself is also unlikely to help us identify meaningful response times that relate to business activity or to track transactions that span multiple systems in distributed applications. This then leads us to the necessity to take a fresh approach to the problem, which is more attuned to the distributed nature of open system environments and applications.

Measuring Business Transactions

To support our ability to achieve the first goal of performance management which is maintaining user productivity, we need meaningful response times that are directly related to the way users perceive the responsiveness of the computing facilities and the way we measure user productivity.

Fundamental to this approach is the concept of business transactions. A business transaction is the complete process the user follows in order to satisfy a customer request or to undertake some task related to achieving the business objectives. The transaction could be as simple as providing an account balance to a customer in a bank or as complex as updating a CAD design for an engineering application. It could be time critical such as a financial trading transaction, and may involve several user interactions with the local system as well as a number of data or processing requests from the local system to one or more remote servers. In the extreme it could involve requests to a system external to the corporation, for example inquiries to an external stock exchange data base. Identifying business transactions has been important for workload forecasting and capacity planning as outlined in [ZAH88].

These business transactions relate to individual applications where an application is defined as a group of processes which together automate or support a group of related activities which are associated with one aspect of the business. To completely understand performance relative to these business transactions, we need some mechanism for identifying the begin and end of each such transaction, and therefore the associated response times and number of these transactions which are completed in a given time period.

With the limitations of currently available measurement technology, the only feasible way of measuring business transactions is to instrument the actual application itself. The application code itself could calculate response times and transaction rates and simply log this information. This will usually not support the allocation of resources utilized by each transaction type, since there will not be markers in the system performance logs to indicate when transactions commenced and completed. This limits our ability to achieve the second goal of performance management, optimizing resource utilizations and hence return on investment in the computing facilities. An alternative is to place calls within the application code to another shared module or perhaps to an operating system intrinsic to denote the beginning and end of each transaction to be tracked, and have a general monitoring facility track these transactions.

There are some specific considerations with this approach, particularly with UNIX-based systems which are process oriented. We need to have a mechanism for determining whether a transaction started by one process can be completed by another, or whether we want the same process to both commence and complete the transaction. We also need to consider the case of a transaction which commences on one system and completes on another (such as a banking transaction which completes when the accounts on the corporate mainframe are updated), and those transactions which start and complete on the same system but which have some client-server interaction imbedded within them. In the latter case, total elapsed time may be available, but the components of that response time may be difficult to identify.

New technology is becoming available which supports capturing identified business transactions in open system environments. System vendors may provide operating system intrinsics to mark the beginning and end of business transactions, which can then be correlated with the other service level and resource utilization metrics captured by the operating system performance measurement facility. Third party performance management software vendors could also provide code modules which could be called to identify transactions, and again this would allow correlation with other information captured by these monitors. Corporations could independently develop their own facilities to at least provide service level measures such as response time and throughput rates for business transactions by instrumenting their application code. This information together with other performance information could be post processed in order to provide some correlation of resource consumption, however this will typically be highly summarized and somewhat inaccurate.

Today's technology as described above will allow us to accurately represent true response times and transaction rates from the user and business perspective, but will not generally allow us to identify components of response time such as network delay and time spent in the server. If the transaction takes place on single system, we may also be able to identify resource consumption for each transaction, however if the application is client-server involving multiple systems, it is very difficult to allocate all resource usage across the systems and network to the individual transactions. Additional research is being undertaken into tracking business transactions across the network, including the ability to identify and allocate response time components and resource utilization by transaction. Approaches under examination include the ability to send the transaction identifier in the header of the remote procedure call and instrumenting the remote procedure call itself to capture response time components.

Monitoring Performance from a Business Perspective

Once we have the ability to identify, capture and measure service levels by business transaction, we can then monitor performance and identify situations where we are failing to provide the required service levels to maintain user productivity. We also have the ability to view performance from the same perspective as the computing facility users which should greatly help in the process of understanding users' concerns and in characterizing perceived performance problems.

The capability to measure performance from the user's perspective supports more effective on-line performance management of open system environments. Instead of waiting for users to be affected by performance problems, or actively monitoring all of the resources and applications within the computing facilities, we can focus on areas requiring attention by looking for deviations from a normal or desired state. This technique, referred to as management-by-exception, generally involves some form of alarm notification based on pre-determined thresholds being exceeded. This concept is particularly important when managing large numbers of systems within a distributed environment where it isn't viable to examine performance on a system by system basis.

Management by exception is implemented by determining the important performance measures to be monitored and the corresponding thresholds which will indicate exception conditions. Then as the performance information is captured and logged, it is examined by an intelligent alarm agent which identifies exception conditions and notifies a central analysis station. Only the alarm notification and supporting information is sent to the central location to ensure the network is not overloaded. By measuring business transactions, we have the most accurate indicator of how user productivity is impacted by any degradation in the performance of the computing facilities, whether the problem is occurring on a particular system or within the connecting network. Because we are focusing only on those areas requiring attention, this technique can also help lower our management overhead.

Depending on the alarm technology employed and how the thresholds are set, this technique can warn of potential problems before users experience difficulty. Instead of waiting for user complaints, we can identify, characterize and correct problems before user productivity is affected, thereby achieving our first objective for performance management. We could identify these potential problems by looking for deterioration in service levels over time or by looking for instances of over utilized resources.

Resource utilizations can generally be captured at a global, application and process level, although certain measurement technologies don't provide application measurement on-line, rather they require process records to be post processed in order to produce application information. It is also important to capture resource consumptions at an individual transaction level. If we determine that a particular application is consuming excessive resource, we need to go further to identify which particular transaction is the problem. This will help us to determine an appropriate cause of action which may involve tuning the application modules which support that transaction or perhaps altering the process such that a background job is spawned to complete that transaction if it is not time critical. By allocating resource usage to individual transactions we can also implement more effective charge back for the computing facilities where the direct costs associated for each business transaction are accounted for. This allows more informed business planning and decision making. As indicated previously, it is possible to allocate resource consumption by transaction if it occurs on a single system, but more difficult if the application is distributed across multiple systems.

Our third objective of performance management is to plan for future capacity requirements based on changing business needs, and this objective is also well supported by adopting the business transaction approach.

Capacity Planning using Business Transactions

The starting point for any capacity plan is to review the objectives of the business, the various user applications and how these support the aspects of the business. We then need to understand the priority, workloads and required service levels of each of these applications. We also need to examine the corporation's business plan to determine how the demand on these applications will change in the future, and what new applications and workloads will be required to meet other future business demands.

If we are identifying and measuring business transaction response times, transaction rates and resource consumption, we have excellent background information for both quantifying current performance and for projecting future growth in terms which are related to the business and which can be directly correlated to computing facility performance. This approach is an extension of the use of Natural Forecast Units which has been utilized for some years. [LO86] [BOW87]

There are various techniques which are used in producing a capacity plan. These include statistical forecasting, analytic modeling, simulation and benchmarking. Statistical forecasting is best used for determining future workload and resource utilization levels on a single node. It utilizes time series analysis to determine future levels based on extending past trends. More sophisticated statistical forecasting techniques also incorporate business units where key indicators of future business activity influence the outcome of the forecast. If we are already tracking business transactions then these can form the basis of our business unit inputs. We can then use the corporation's business plan to project future business transaction volumes, and as we have historical data relating business transactions to both service levels and resource consumption, we can confidently forecast future performance and resource utilization levels.

Analytic modeling has been predominately used to predict future performance of computer systems. This approach builds a mathematical model of the system utilizing mean value analysis, and can answer "what if" questions such as "what happens to response time if I add ten more users to a specific application?" The analytic approach has not traditionally been used to model the performance of network configurations. Again here the use of business transactions enhances this approach as we define our workloads by these transactions. We know resource consumptions by transaction and can validate the base model by comparing predicted performance with measured performance for each transaction. This then allows us to conduct "what if" analysis based on varying the number of business transactions, number of users or some aspect of the system configuration.

Simulation has been a popular method for modeling performance of networks. It provides a representation of the environment, and simulates the flow of each transaction and event through the networked computing environment. Simulation takes more resources and a longer time to produce a result than does statistical forecasting or analytic modeling, but can be used to provide more accurate results for complex situations. The ability to identify, track and simulate business transactions will enable a more realistic and accurate picture of future performance based on projected changes within the business.

Modeling distributed environments is a developing area for both expertise and tools. Most capacity planning products to date have either focused on the network or the individual systems within the network. To effectively predict future performance of distributed applications, we need to model both the network and system elements. We also need to be able to track and measure client-server transactions across the distributed environment in order to validate these predictions. This is where utilizing business transactions as the core performance metric can be invaluable as a way of understanding transactions with client-server open system environments. Much work is currently underway in the area of capacity planning distributed environments, and some of the previously mentioned research into tracking transactions across the distributed environment will be required for accurate distributed environment modeling.

Conclusions

The open system environment requires a fresh perspective on the definitions of response times and transaction rates. The traditional definitions assume a terminal device accessing a central system and are not consistent for different UNIX-based systems and applications, nor do they support distributed client-server type applications. Identifying business transactions within applications provides an excellent (and with the limitations of today's measurement technology the only effective) method for accurately monitoring computer facility performance from the business and user perspectives.

Once this step of instrumenting applications has been undertaken, the benefits are great in terms of supporting the three objectives of performance management. We have the same view of performance as do the users, and hence we are more effectively able to identify situations which will lead to a loss of user productivity. When we do experience performance problems, we will be able to discuss issues with users in familiar terms related to the business applications, and hence characterize the problem more easily.

We can also use the business transaction perspective when we optimize resource utilization and manage the capacity of our computing facilities. By looking at the resources consumed by individual business transactions, together with a knowledge of the relative importance of each transaction, we can take more informed decisions regarding tuning opportunities or workload balancing if we become constrained on available resources. Similarly, projecting future workloads for capacity planning exercises is made easier by using business transactions, as our workload projections can be derived directly form the business plan and by discussing future requirements with facility users in business terms they understand.

Today, the business transaction approach can be implemented either by system vendors providing operating system calls, by performance management software vendors providing code modules which identify and log business transaction information, or by corporations taking the initiative and developing such facilities in house. Additional research is being undertaken by several vendors in the area of tracking transactions across the network, and this work should result in more sophisticated resource consumption information, which in turn will assist in more accurate capacity and configuration planning and management.

References

[LO86] Lo, T.L. and Elias, J.P., Workload Forecasting Using NFU: A Capacity Planner's Perspective, 1986 CMG Conference Proceedings, Computer Measurement Group

[BOW87] Bowerman, James R., An Intorduction to Business Element Forecasting, 1987 CMG Conference Proceedings, Computer Measurement Group

[ZAH88] Zahavi, William Z. and Bouhana, James P., Business-Level Description of Transaction Processing Applications, 1988 CMG Conference Proceedings, Computer Measurment Group

Michel Ferachoglou - HP

Meeting customer security requirements: a cryptographic security module for the HP 9000

Numerous important trends are dramatically complicating the challenge of securing an organization's information resources. For example, the increasing competitive pressure for an organization to share and access data securely on a global basis; the need to improve organizational efficiency and productivity by decentralizing the computing power from large data centers to the user desktop in a client/server paradigm; the adoption of an open systems strategy to cost-effectively distribute computing and information; and the growing threat of hackers.
HP has a good understanding of the global enterprise-wide security problem and thus, has developed the Secure Open Computing strategy. We think that this will maintain our leadership in the area of open systems and distributed computing solutions.

Crypto devices and associated services are key components of the Secure Open Computing framework: Encryption Cards provide a tamper-resistent environment for the implementation of security services such as data confidentiality, data integrity and message non-repudiation via cryptographic means; Smart Cards provide a tamper-resistent environment for the implementation of strong end-user authentication.
Applications are many (secure messaging, secure EDI, software distribution, etc...). Security modules will play an important role toward ensuring the success of client/server deployment by providing efficient high-speed end-to-end encryption services integrated under DCE.

The session will briefly review the evolving security needs of businesses, highlight the benefits provided by Encryption Cards and Smart Cards, and explain their position within the HP Secure Open Computing. As an example, the audience will be presented a banking application implementing Electronics Funds Transfers that takes advantage of the Cryptographic Security Module for the HP 9000 Series 800. This device offers a high degree of trust, far beyond that which can be placed in computer operators, applications or even the main operating system software.

PAPER NUMBER: 2012

TITLE: Giving End Users Access to Corporate Data

PRESENTER:
Ron Zambonini
Cognos
3755 Riverside Drive
P.O. Box 9707
Ottawa, Ontario, K1G 3Z4
CANADA

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Janet M. Muto
Hewlett Packard 300 Apollo Drive
Chelmford, MA

SETTING A DESKTOP STRATEGY

The roles of computing and of the IT Manager have gone through a tremendous change. There has been a significant series of events that have caused this to happen. Global competition and opportunity, more savvy customers, significant cost pressures on budgets and the literal explosion of technologies have changed both the role of IT and its place in the organization forever.

Executives are seeking out ways for their companies to become more competitive and are viewing IT as a potential competitive edge. Most importantly, the requirement to know your customer has created a whole new series of needs that require that organizations think differently about how they are organized and how they track information.

All of these requirements have landed broadly on the shoulders of the IT manager who must now wrestle with both the legacy environments created in the years past and try to meeting the challenge of keeping the organization competitive through it use of information.

In most cases, businesses are looking at re-engineering their business processes as well as their systems. In fact, recent articles (ie. June 14, 1993 Business Week on the *Technology Payoff*) have indicated that companies who benefit from white collar productivity gains due to the use of IT are the companies who have re-engineering their business processes as well.

We believe that the ability to change and the management of that change is the key differentiator of the most successful businesses. At Hewlett Packard, we are constantly changing to meet new market requirements, but we also recognize that constant change does not mean thrashing. Rather, the management of the change and the flexibility of the environment to embrace change should be built into every companies' strategy and underlying system architecture.

Programs like our Mainframe Alternative Program which assist our customers in either moving to a new architecutre or in providing for new technologies that still work within the legacy environment have been extremely successful because they deal with the overall strategy (people, networks and systems). Once a company has decided to make a change in their overall system and network architecture we find that many are opting to implement new systems with a client/server architecture and we have teams available that will assist with that process. One of the key decisions that arises out of the client/server strategy is what is the future direction and plan for the organization and how do I choose the right desktop to meet both my current and future needs?

We believe that the decision of what to put on the desktop to assure the right desktop for the right job both today and tomorrow is critical. We are all on a path that will lead us to object technology within the next few years and we need to ensure that our current architecture will get us there. In the best of circumstances, we want to be able to provide as little change to the user's interface as possible while continuing to add to the capabilities of that desktop and benefits that its users derive from it. It is critical that we understand what the ongoing requirement of a user (or class of users) will be. Once those are defined we can begin to segment the users and their desktops.

Overall we believe that there are 3 major segments within the desktop market. These can be characterized as the traditional engineering/scientific desktop which we characterize as the **Design Desktop;** the productivity desktop which we characterize as the **Personal Desktop;** and a newly emerging category of use who needs the integrated performance, multi-tasking and communications of the workstation without compromising the familiarity, application legacy and price points of the Personal desktop! We categorize this as the **Enterprise Desktop.**

There are certainly different platforms that are associated with these classes of users and often they often overlap. Today, I will review the examples of these platforms and finally spend more time developing the use and user of the Enterprise Desktop along with the needs of that segment.

Today's choices of desktop platform are varied. At Hewlett Packard for example, we have a wide range of Intel based Vectras, a newly announced terminal replacement product called the Windows Client, and a leading family of Xstations and workstations. One of the questions that I am asked most often by the IT management that I present to is how determine the right desktop for the right job given all these choices.

Perhaps the best place to start is with the Design User. Obviously the workstation has been the traditional choice in this space, with the use of Xstations on the design desks where local performance is less of an issue, but the need to window into the design environment and to collaborate is critical. With each decrease in the price points of both workstations and Xstations we see a significant increase in the number of users who take advantage of workstations and xstations.

The design user is characterized by their need for high performance on the desktop and a high level of workgroup communications. This user generally accesses only one or two databases but then spends considerable time and effort on computing and manipulating that information. This user is also characterized by the generally numerous simultaneous operations that they spawn.

The increasing sophistication of the tools available to the Design segment requires greater and greater performance and sophistication in the platform that they use. Certainly Risc based Unix platforms will continue to dominate this space for both the local performance the Risc based platforms allow and for the distributed windowing and integrated graphical capabilities of the Unix Operating environment.

The second area of computing on the desktop which I mentioned is what we call the Personal desktop. This is the desktop/user who is primarily accessing one application, like the HR system or the accounts payable system and may also need to use traditional productivity tools like word processing, mail, spreadsheet, presentation graphics etc. This user is characterized by low levels of local computation, data manipulation and simultaneous activities. This user also spends most of his/her time working with information that resides either on their one application (which they download) or on the local productivity tools resident on their desktop.

This segment of users is best served by a low cost client which offers the benefits of a terminal (for ease of administration and management by IT) and offers the local productivity of the PC. In Hewlett Packard's case, we offer several desktops that fit this classification: the full line of Vectra based PCs in a networked environment certainly will appeal to the users who do more local computation and manipulation, while our newly announced Windows Client and our leading family of Xstations allow use of productivity tools and application performance with the benefit of being easier to administer than PCs or workstations for that matter.

The final segment is the area which I will spend the most time one: this is the user that needs what we are calling the "Enterprise Desktop". The "Enterprise Desktop" reflects not a product, but a set of characteristics that separates than user from both the Design and Personal user.

The Enterprise Desktop segment is characterized by the time critical nature of the jobs they perform and of enterprise wide implications of the decisions that they make and/or the customer they contact. Generally, the workgroup that supports this user is global and cross functional and therefore the supporting technology requirements are global in nature as well: i.e.. high performing networked systems. The application environment that this user works in tends to be different as well.

Generally this user is characterized by the combination of information and resources necessary for his/her decision making. Whereas the technical user is characterized by heavy use and computation of information from one or two applications, and the personal user is characterized by use of multiple personal applications along with access to one or two server based applications, the Enterprise user is one who accesses many databases and applications simultaneously, computes and manipulates that information and uses output from one source as input to another.

This user's need for enterprise wide information and for manipulation of that information is time critical and allows that individual to make decisions quickly and effectively. This user's decisions are mission critical and directly related to the revenue of that corporation. Examples of this type of user are Financial Traders, Customer Service representatives (both sales and support), Brand Managers in the Packaged Goods and Retail sectors, and financial analysts/business managers cross industry.

Until now the user which we characterize as the Enterprise User had to choose between desktop technologies and either sacrificed the benefits of the "other" desktop or put both on his/her desktop. An example of this is the financial trader who adopted workstations early because of the high performance and multi-tasking requirements of their application but who had to also put a PC on the desk to do his/her personal applications. On the other side, there are many users who have opted for networked PCs with Windows who are suffering from lack of balanced performance and true multi-taking and distributed windowing capabilities necessary to do their jobs.

Today, more and more users are Enterprise users and need to have the capabilities listed previously. They do not want a desktop that is just a vehicle to ACCESS information (as many have deployed Client/Server today) but rather to be an active and integral member of the information architecture. The "battle" for the Enterprise Desktop will play out between PC world moving up to new hardware and OS technologies and the traditional workstation technology moving to commoditize Unix and run new OS environments as well.

In order for either of these two technologies to be successful and to truly serve the Enterprise Desktop customer we need to ensure that the desktop meets the following criteria:

(1) Enterprise System and Network Management
(2) Application Availability
    Personal/Shrink Wrap
    Application Development Tools
    Mission Critical
(3) Integrated Communications
    Integrated LAN, Wan and Remote networking
    Multimedia communications
(4) Intuitive
(5) Low Cost

At Hewlett Packard we are working to meet and exceed the baseline requirements in all of these areas as they pertain to the Enterprise user.

Specifically, as a company we have dedicated an entire division to solving the problems that large IT organizations face in trying to offer the benefits of distributed processing to their users while managing and serving that environment like a mainframe! Our Openview product is an industry standard network and systems management product and we are continuing to innovate in this area. In addition, we are porting or have available all of the other leading Enterprise systems management products available or announced on our server and desktop platforms.

We are also looking at ways that we can integrate our offerings to our users with a much more intuitive and natural GUI (standards based of course) and to make it easier to manage and secure information in distributed environments.

The second key area is the availability of applications: The Enterprise desktop user needs to have a breadth of applications that span areas such as ERP, financial trading, statistical analysis, distribution etc. and provide access to the world of shrink wrapped software currently available only in the personal world. In addition, most of the users of the Enterprise Desktop require a high level of involvement of their IT departments to provide information access and update, security and even a customized environment. Finally, the environment must be able to provide as if PC (current standard 486) application availability and performance and even Mac application availability all on the same desktop.

The next area that is critical to the Enterprise Desktop user is in the area of communications. This covers two areas: the physical computing communications between the various systems and architectures intalled and the intuitive communications from user to user. The Enterprise Desktop user needs the integration of audio, video, telephony and workgroup sharing capabilities to better do his or her job. The user must also have easy access to legacy data and cross functional information and then be able to manipulate it with personal tools and print as if he/she is on the PC LAN.

Finally, last but not least, the desktop must be familiar, intuitive to use and have a low overall cost -- both in terms of the capital investment and long term cost of ownership. HP has innovated with products like HP Vue which is now the basis for the Cose Desktop Environment. In addition, products which continue to further this area: intuitive, easy to manage and custom tailor "able" will be announced within the next few weeks.

In terms of the cost of the desktop, I believe that all of you in the audience understand what is happening to the cost of technology. Certainly, lower cost and higher performance are now standards of all of our business plans and our next year's plans are no exception.

So, now that you understand the Enterprise Desktop concept, what technology can support those needs? This user must have an architecture that supports his/her requirements. Unlike the traditional mainframe/terminal model of computing, or even the way that many have implemented client/server architectures -- with PC accessing a large mainframe-like system, the Enterprise User need to be an integral part of a distributed environment. This user is not just accessing information but using the output from multiple data sources as input to the next series of decisions. In addition, this architecture must be capable of allowing change and flexibility. In many of the examples mentioned, the ability to change the system to meet changing needs can allow for greater revenues and profits.

We believe that today the technologies available for the Unix workstation and the new and ever lower price points make it the ideal Enterprise Desktop. Certainly in the future new OSes will offer some of the same characteristics, especially when available on Risc workstations.

However, today Unix is the only real answer. It offers a true pre-emptive multi-tasking operating environment, supports advanced multi-vendor networking, allows true distributed windowing capabilities and with technologies like our announced WABI port you can run most of your personal applications, emulate a Mac and operate in an environment that is less expensive to support and manage than a PC -- all on your Unix workstation. In fact, our internal studies and customer information corroborate the fact that the cost of ownership of a Unix workstation is significantly less than a PC despite the differences in their initial capital costs!

So the decision of what technology to place on each person's desk is not an easy one. However, if you break up the person's job into a series of measurable activities and potential future activities, you will determine that there are different groups and functions of users within your organization whose technology and business needs are different. Your goal should be to work with your vendor (hopefully Hewlett Packard) to ensure that your choices leave open room for change and innovation.

The easy way out is to treat all users the same and to drive to a common hardware and/or desktop platform. However, today, there is no one answer that will provide the right solution across the entire spectrum of users (despite what some companies are saying)! Your decisions need to take into account the uses and users of the desktop and the various hardware, software environments, applications and tools available. In addition, you should really test the ease of management of the chosen environments. Often times the benefits of system management, scaleability and cost of ownership are lost to the immediate goal of lower capital costs

As the IT management of global businesses it is your obligation to provide the best suited desktops to the users and to also provide an environment that is not too difficult to manage. With new environments becoming available on multiple platforms (NT and Next for example) and with efforts like the Common Operating Software Environment and CDE initiative the desktop and its role in the overall "system" is gaining more and more importance. Comapnies that recognize this and use this to add value to their mission critical users will ultimately be more competitive and profitable.

PAPER NUMBER: 3001

TITLE: Developing Cross-Platform Multimedia Applications Using Icon Author

PRESENTER: Leo Lucas
AimTech Corporation
20 Trafalgar Square
Nashua, NH 03063-1973
603-883-0220

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

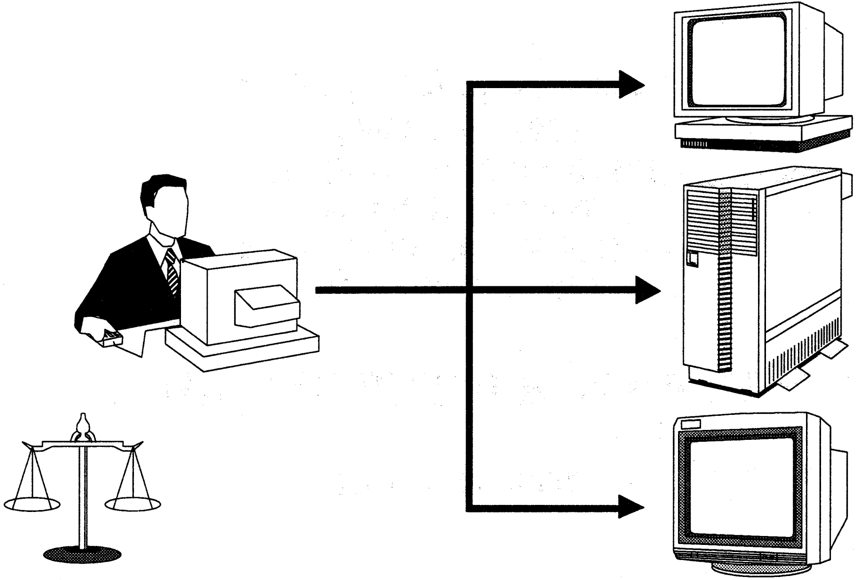
Paper No. 3002

HP Task Broker Release 1.1

by Renato Assini
Member of Technical Staff
Hewlett Packard
Chelmsford Mass, 01824
(508) 256 - 6600
assini_r@apollo.hp.com

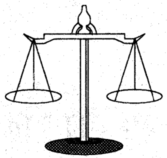
HP Task Broker

Simplified Access to Compute Resources



HP Task Broker

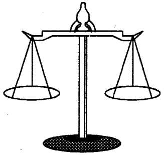
Agenda



- Introduction of the Product (30 Min.)
  - Prior to Task Broker
  - Features of Task Broker
  - How Task Broker Works
- Features of Release 1.1 (20 Min.)
  - Group Configuration
  - Configuration Editor
  - Configuration Manager
  - Task Manager
  - Online Context Sensitive Help

Prior to HP Task Broker

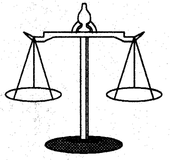
Networked computer users:



- Gathered and analyzed machine specific information
- Selected the most available server
- Connected to the selected server via telnet, remsh, or crp
- Copied program and data files to the selected server via ftp or NFS
- Invoked applications over the network
- Copied resulting files back from server via ftp or NFS

HP Task Broker

The Product



Selects best network resource based on availability, processing power, and loading

- Distributes computational tasks among heterogeneous UNIX-based computers
- Requires no changes to the application
- Enables access to more compute power when needed, and reduced hardware costs
- Transparently locates the most available server for a computational task

Resulting in . . .

HP Task Broker
The Solution

. . . maximum throughput and productivity through

- **Efficient Job Distribution**
- **Load Balancing**
- **Transparent Data Access**
- **Fault Tolerance**

thus utilizing the full potential of your network

HP Task Broker Features

Features (1)

- Automatic server selection for job/application
  - Selection process transparent to user
  - Location of service need not be known
- Work assignment based on "bidding" by servers
- Flexible bidding algorithm considers time of day, current CPU load, disk space, etc.
- No application changes required

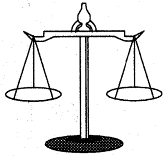
HP Task Broker Features

Features (2)

- Users can control which machines are servers, and when they are available
- Runs in a heterogeneous environment
  - HP-UX
  - Domain/OS
  - SUN/OS
- Other machines can act as servers via surrogate
- Heterogeneous machines can interoperate and exchange work load

How Task Broker Works

A Chronological Description (1)

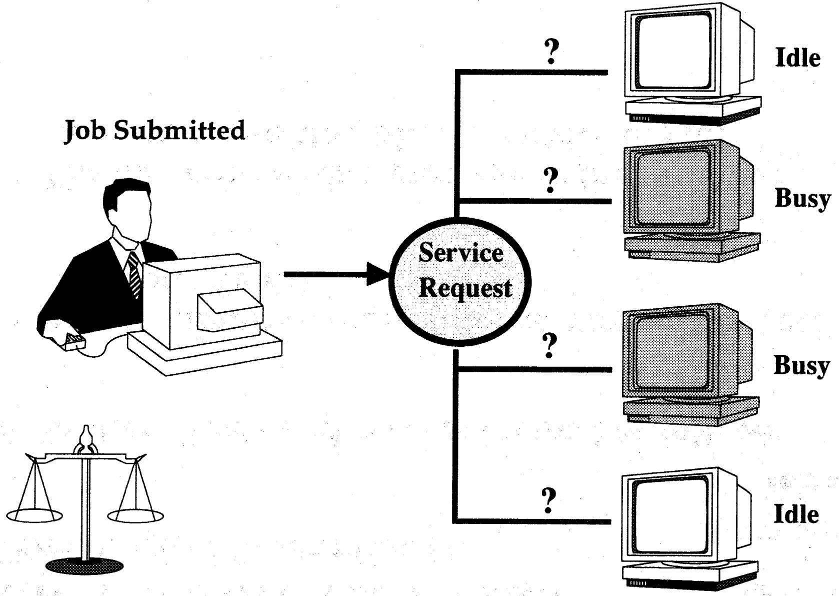


Task Broker clients and servers interact as follows:

- A user submits a request for service to the local daemon (client)
- The daemon sends a message to the group of servers, requesting bids to service the job

How HP Task Broker Works

A Chronological Description (2)



Job Submitted

Service Request

?

Idle

?

Busy

?

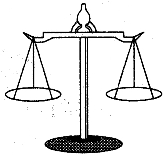
Busy

?

Idle

How HP Task Broker Works

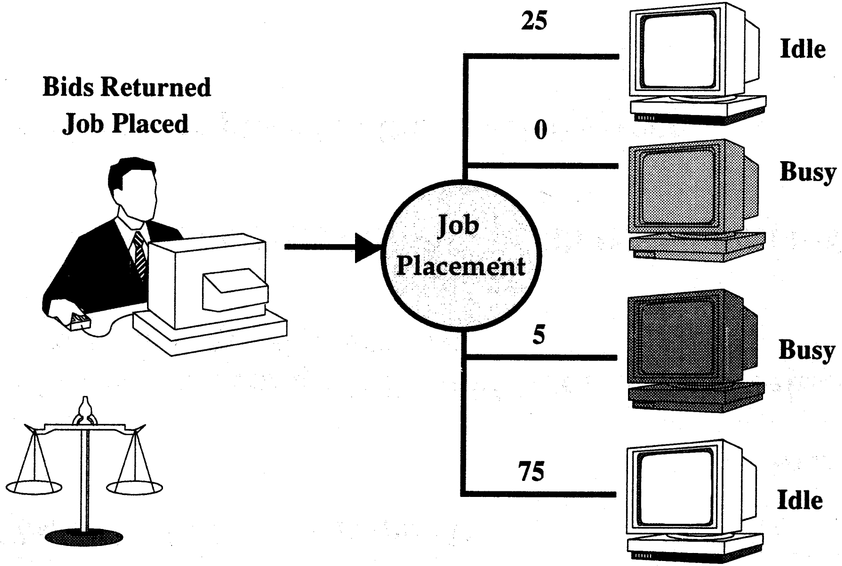
A Chronological Description (3)



- The servers compute their bids, or "affinity values", for the requested service
- The client selects the server with the highest bid
- The job is placed at the selected server

How HP Task Broker Works

A Chronological Description (4)



Bids Returned
Job Placed

Job Placement

25 Idle

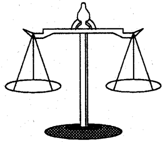
0 Busy

5 Busy

75 Idle

Features of Task Broker 1.1

Group Configuration



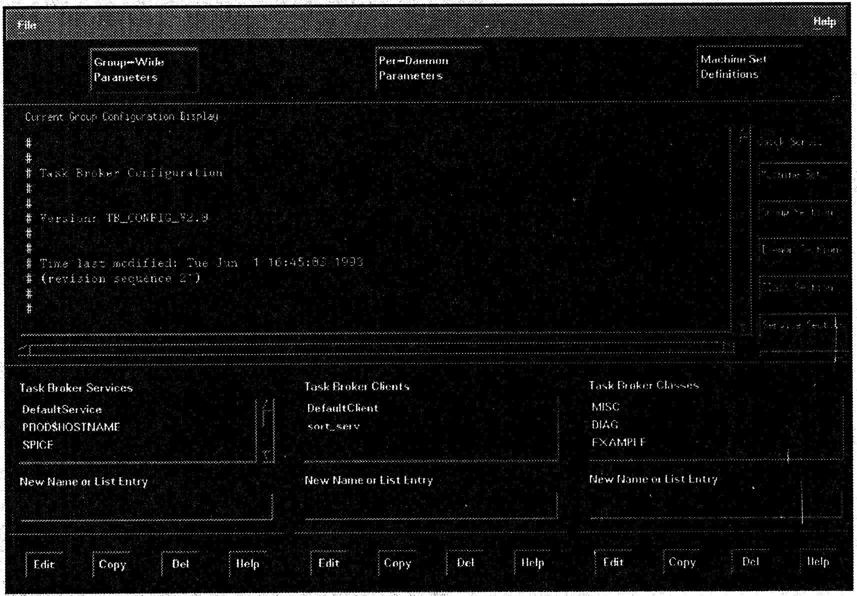
- Single centralized configuration file for the entire Task Broker Group
- Replaces per-machine configuration files of previous versions
- Employs master/replica scheme preventing single point of failure
- Edited with an integrated forms based editor

Features of Task Broker 1.1

Configuration Editor

- Graphical interface to edit group configuration file
- All configuration parameters are displayed
- Eliminates syntax and semantic errors
- Displays entire group configuration
- Optionally displays a machine resolved configuration

Task Broker Configuration Editor


File
Help
Group-Wide Parameters
Per-Daemon Parameters
Machine Set Definitions
Current Group Configuration Display
Task Broker Services
Task Broker Clients
Task Broker Classes
New Name or List Entry
Edit
Copy
Del
Help

```
#
#
# Task Broker Configuration
#
#
# Version: TB_CONFIG_V2.0
#
#
# Time last modified: Tue Jun  1 16:45:03 1993
# (revision sequence 27)
#
#
```

Task Broker Services:
- DefaultService
- PROD$HOSTNAME
- SPICE

Task Broker Clients:
- DefaultClient
- sort_serv

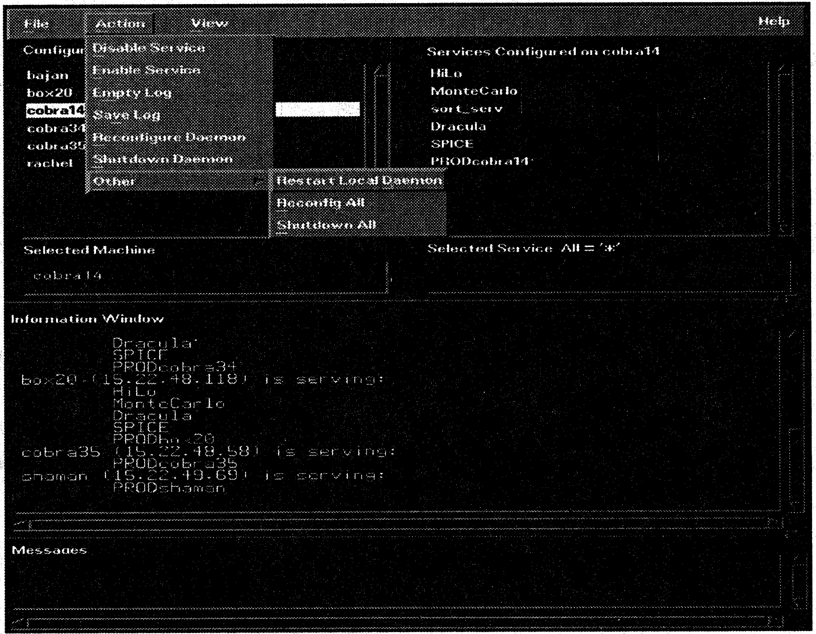
Task Broker Classes:
- MISC
- DIAG
- EXAMPLE

Features of Task Broker 1.1

Configuration Manager

- Enables single point management of local and remote Task Broker daemons
- Enable and disable Task Broker services
- Empty and/or save Task Broker log files
- Reconfigure and/or shutdown Task Broker daemons
- Obtain information on the state of Task Broker daemons
- Start the Configuration Editor
- Must be root (or in tbroker group) most operations

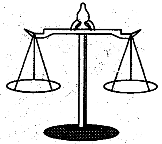
Task Broker Configuration Manager


File
Action
View
Help
Configur
Disable Service
Enable Service
Empty Log
Save Log
Reconfigure Daemon
Shutdown Daemon
Other
Restart Local Daemon
Reconfig All
Shutdown All
bajan
box20
cobra14
cobra34
cobra35
rachel
Services Configured on cobra14
HiLo
MonteCarlo
sort_serv
Dracula
SPICE
PRODcobra14
Selected Machine
cobra14
Selected Service All = '*'
Information Window
Messages

```
              Dracula
              SPICE
              PRODcobra34
 box20 (15.22.48.118) is serving:
              HiLo
              MonteCarlo
              Dracula
              SPICE
              PRODbox20
 cobra35 (15.22.48.58) is serving:
              PRODcobra35
 shaman (15.22.49.69) is serving:
              PRODshaman
```

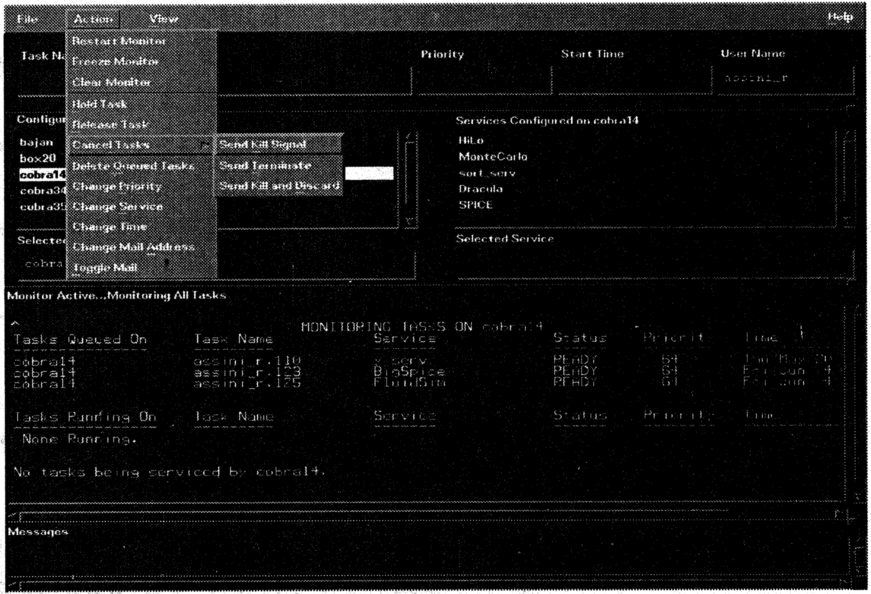
Features of Task Broker 1.1

Task Manager



- Graphical interface to monitor and modify Task Broker service requests (tasks)
- Allows users to monitor tasks on local and remote Task Broker machines
- Can modify queued and running tasks submitted from the local machine
- Able to display detailed task state information
- Can monitor all tasks, tasks by a given User or tasks for a specific Service

Task Broker Task Manager


File
Action
View
Help
Restart Monitor
Freeze Monitor
Clear Monitor
Hold Task
Release Task
Cancel Tasks
Send Kill Signal
Send Terminate
Send Kill and Discard
Delete Queued Tasks
Change Priority
Change Service
Change Time
Change Mail Address
Toggle Mail
Task Na
Priority
Start Time
User Name
Configur
Services Configured on cobra14
Selecte
Selected Service
Messages

User Name: assini_r

Configur:
- bajan
- box20
- cobra14
- cobra34
- cobra35

Services Configured on cobra14:
- HiLo
- MonteCarlo
- sort_serv
- Dracula
- SPICE

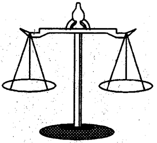
Selecte: cobra

Monitor Active...Monitoring All Tasks

```
                    MONITORING TASKS ON cobra14
Tasks Queued On    Task Name       Service     Status   Priorit   Time
cobra14            assini_r.110    x_serv      READY    64        Thu May 20
cobra14            assini_r.123    BigSpice    READY    64        Fri Jun 4
cobra14            assini_r.125    FluidSim    READY    64        Fri Jun 4

Tasks Running On   Task Name       Service     Status   Priority  Time
 None Running.

No tasks being serviced by cobra14.
```

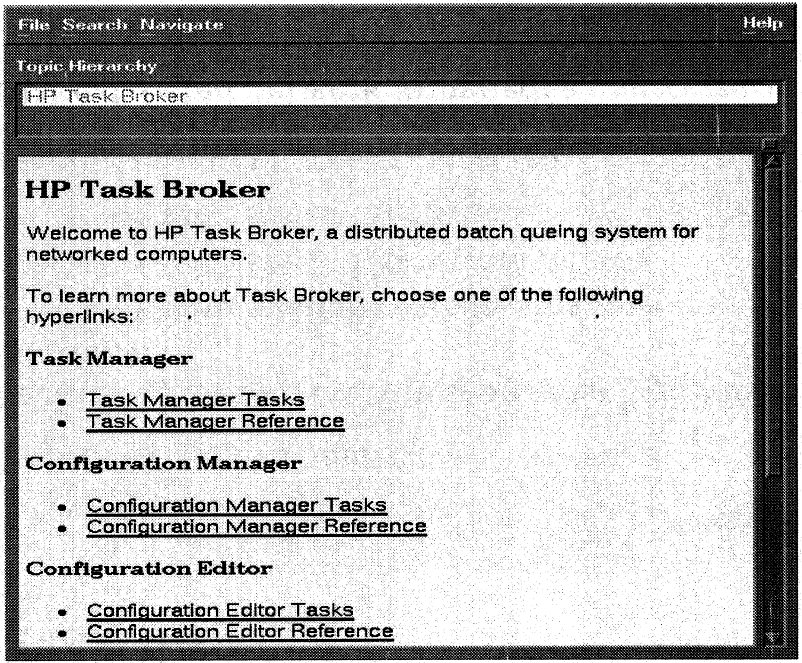
Features of Task Broker 1.1

Online Help



- Online access to information on all graphical interfaces
- Incorporates the HP Help System
- Context sensitive help is available at your fingertips
- Hyperlink capabilities guides you to related topics
- Task Manager manpages available
- Imbedded Task Broker tutorial

Task Broker Online Help


File Search Navigate
Help
Topic Hierarchy
HP Task Broker

HP Task Broker

Welcome to HP Task Broker, a distributed batch queing system for networked computers.

To learn more about Task Broker, choose one of the following hyperlinks:

Task Manager

- Task Manager Tasks
- Task Manager Reference

Configuration Manager

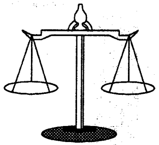
- Configuration Manager Tasks
- Configuration Manager Reference

Configuration Editor

- Configuration Editor Tasks
- Configuration Editor Reference

HP Task Broker 1.1

Summary

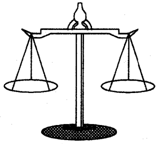


With Task Broker one can...

- Improve *time to completion*
- Facilitate program execution on *most "appropriate" machine:*
    - with available processing power
    - with special hardware devices
    - with special software
- Enablework to be done in *parallel*
- Make detailed network knowledge *unnecessary*

HP Task Broker 1.1

Summary



Task Broker solves problems by providing:

- A *truly distributed solution* - providing equal access to heterogeneous compute resources for all users
- *Graphic User Interfaces* - greatly simplifying ease of use

- A development and support organization *committed to providing the best solution* for our customers' distributing computing needs

PAPER NUMBER:	3005
TITLE:	Successful Implementation of Software Configuration Management
PRESENTER:	Tom Burdon Softool Corporation 340 South Kellogg Ave. Goleta, CA 93117 805-683-5777

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

COBOL: The Move To The Desktop.

Colin Bodell

Micro Focus Inc.
2465 East Bayshore Road
Palo Alto, CA. 94303

cib@mfocus.com

Abstract

There is a myth that COBOL is the language for mainframe application development only, and C is the language for developing applications on personal computer systems running DOS, OS/2 and UNIX.

This paper examines some of the considerations involved with moving commercial applications from mainframes and mini-computers to the desktop. Development trends, performance factors, alternative development tools, and support of mission critical data processing applications are all discussed.

It will be seen that, as on the mainframe, COBOL is particularly well suited for the design, development, execution and maintenance of commercial applications that will run on a multiplicity of desktop workstations, networked PCs, and multi-user server systems.

1. Introduction

The value of computer systems installed worldwide continues to grow year on year. In 1986, the worldwide systems market value was placed at $95 billion. In 1991 it had grown to $186 billion, and at the current rate of growth, it is projected to reach $304 billion by 1996 [1]. Single-user systems represent a significant and increasing share of the system market. Personal computers and workstations are winning the battle for the desktop.

IDC reports that 56 percent of all programming worldwide is performed in the COBOL language. This figure represents all classes of systems, but is mostly characterized by mainframe and minicomputer installations. The existing investment in COBOL applications is approximated at $70 billion worldwide, representing over 700 billion lines of code. In order to preserve this investment, COBOL compilers and development tools continue to evolve. The deployment of commercial applications can take advantage of the significant price/performance of intelligent desktop workstations. This paper examines the reasons for the continuing dominance of COBOL as the language of choice for commercial application development, reviews a number of considerations relating to the downsizing of mainframe applications, and discuses a COBOL-based approach to client/server commercial application development.

2. COBOL for Commercial Applications

The criteria for selecting an easily maintainable language for the development and support of commercial applications are the following:

The language chosen for the development of commercial applications must be designed specifically for that purpose. It must contain a wealth of features suitable for commercial use. The language should support "state of the art" structured programming capabilities and well-defined standards. The language should be established enough that it will be available for the life of the application and flexible enough to add enhancement features. COBOL meets this definition. Thirty years ago, the COBOL language was designed and developed to satisfy the needs of the data processing community. Today, COBOL continues to fulfill the requirements of the commercial application developer.

COBOL was designed to be particularly strong in the areas of character handling and data presentation to allow clear and concise specification of the fields that make-up file records. The creation and manipulation of files of character strings makes COBOL ideal for business data processing. A wide range of advanced facilities are built into the language. These include indexed file handling, sorting and merging of data files, and report generation. Specialized syntax for interactive screen and keyboard handling, along with the tight coupling of COBOL to DOS, OS/2 and UNIX operating systems, ensure that business applications take advantage of the immediacy of minicomputers and microcomputers. At the same time, business applications have all the power offered by COBOL on the mainframe.

COBOL is an established language. Eighty-three percent of Fortune 500 companies use COBOL to develop their software applications.

COBOL has the benefit of experienced programmers. Recent surveys estimate that there are over three million COBOL programmers worldwide.

COBOL is easy to learn. Program source code is largely self documenting and recent implementations of the language are highly competitive in raw processing power when compared to 4GLs.

COBOL is a non-proprietary language. Standard versions have been published by the American National Standards Institute (ANSI) since 1968. The 1968 ANSI Standard was refined in 1974. In 1985, ANSI updated the COBOL definition to include structured programming techniques, such as the in-line perform statement that permits a loop to be written without resorting to GO TOs and paragraph names. Further recent enhancements have added intrinsic functions into the COBOL language.

The massive investment in existing applications and commitment by the industry to develop future applications in COBOL indicates that COBOL is here to stay. COBOL, however, is not resting on its laurels. The large COBOL installed base is not static. As corporations move from centralized mainframes to distributed computing, COBOL is evolving with the industry. It remains the leading language for business application development by offering support for new technologies as they emerge.

3. COBOL Futures -- Object Orientation

The future of COBOL is directed by industry standards organizations and compiler vendors. New features, proposed by compiler vendors are ratified, modified and tuned for publication as officially defined standards.

One recently proposed standard is the addition of object- oriented features in COBOL. At this time Object-Orientation (OO) COBOL has backing from a number of compiler vendors and hardware manufacturers, including IBM, Hewlett-Packard and Micro Focus. There are two major concepts underlying object-orientation. The primary premise of object-orientation is that software should be built from reusable components. Components that have been thoroughly tested and can be relied upon to function correctly when employed in a new application. Secondly, because object-oriented roots are in computer simulation that needs to model the real world as closely as possible, an object-oriented programming language must be able to accept and work with data types introduced by the user. Language extensibility is the feature that allows a user to tell the language what data types they are going to use.

A user-defined data type in object-oriented terminology is known as a 'class'. Classes comprise hidden data items, procedural code to manipulate the data items (these procedures are called 'methods') and a published user interface. An 'object' is a copy of a class in an application program.

Object-orientation in COBOL is currently under discussion by the OO COBOL CODASYL working group. A number of compiler vendors are active in formulating a common syntax to bring object-oriented features to COBOL and to ensure the longevity of the language and the applications written using it. Micro Focus has released an OO option to COBOL on the PC as a stepping stone to the implementation of full OO capabilities.

4. Multi-Platform Downsizing Considerations

COBOL has been traditionally dominant in mainframe programming. However, over the past few years the distinct differences between microcomputers, minicomputers and mainframes, have begun to blur.

The processing power levels of these three distinct computer systems (and distinct vendors) have begun to overlap. Traditional mainframe applications continue to be passed down and distributed to work groups and individuals. As the PC continues to build from the departmental level and become more and more connected, the Local-Area Network (LAN) is emerging as the new model for application deployment. In the future, connectivity to, and interoperability with, larger systems will be based on multi-platform, multi-vendor environments as open systems strategies continue to be developed by hardware manufacturers.

The power of the intelligent workstation can be used perhaps most effectively to integrate multiple applications and systems on multiple platforms, and to present to the end-user a single, integrated image of what appears to be a single system. The desktop system, as an end-user computing and communications device, enables the user to access data easily in a heterogeneous, multi-platform network environment.

While standards are beginning to take some primitive shape, the industry as a whole has a long way to go to agree upon a set of standards for building applications. The need by hardware and software vendors to innovate and differentiate by adding proprietary features to a product range will continue to result in standards fragmentation. Even if such universal standards were available today, and all applications built from now on were built on that standard, the weight of existing legacy systems is so vast that it will be well into the next century before even a small percentage of those systems have been converted.

The future of desktop computing is not, however, limited to technology. The human, organizational factor is equally important, as it is the Information Systems (IS) organizations that must guide their companies through the jungle of computing opportunity.

5. IS Role

Users must realize that information and other Information Technology (IT) assets must be protected and managed. They should also realize that they cannot yet solve all the enterprise application needs with PCs or LANs.

Many companies are addressing these issues by realigning their IS organizations according to function rather than technology platforms, by aggressively incorporating PCs into the IT architecture, and by moving to next-generation systems to reduce the enterprise cost of computing and increase group productivity.

The challenge facing IS organizations is how to implement a broad range of application architectures to fully exploit the PC platform. In addition to monolithic server-based applications, IS must implement client/server and monolithic PC-based applications.

Effective utilization of the intelligent workstation requires IS to do more than supply users with off-the-shelf PC applications and terminal emulation. Just moving all or part of a host-based application will not significantly reduce the application backlog. IS must focus on empowering end-users, both through corporate development of network applications and through delivery of client-side tools and applications, and back-end services.

In order to help the move from mainframes to micros and networks, IS will need to learn PC skills. Micro Focus tools help bridge this knowledge gap by providing development tools within the context of the familiar mainframe tools, with COBOL at the core of the technology. As well as the traditional editor/compiler/debugger array of components, COBOL based tools are presented in a number of ways.

CASE

CASE tools including Life-Cycle CASE and Integrated CASE (I-CASE), benefit from the ability to work with, and emit as necessary, standard COBOL. I-CASE solutions are proprietary in development, but some are moving towards the generation of "open" code. For example TI's IEF will cross-generate COBOL and C.

It is noteworthy that, in a Gartner Group survey published in August 1991, among 30 CASE vendor products, Micro Focus was the most commonly used by those users participating. Micro Focus stood out with a high number of users, and not one declining to purchase the product again.

Integrated Development Environments

The Micro Focus COBOL Workbench is perhaps one of the most significant renderings of an application development environment available today. UNIX and open systems bring with them their own needs for integrated development environments. Extensibility, the ability to integrate the development environment with end-user developed and third party tools, is a primary feature of such tools as IBM's Workbench/6000 and NCR's Application Development Environment. Both of these environments support COBOL compilation and debugging tools supplied by Micro Focus. These development environments combine the connectivity of the integrated desktop, with the GUI "friendliness" of the individual tools. Developers become much more productive when individual and inter-tool access is optimized.

4GLs

It is important that IS organizations examine alternate methods of application development and maintenance. One set of alternatives to COBOL as a business application language, are Fourth Generation Languages (4GL). While 4GLs claim to write, debug, and maintain code easier and faster than a Third Generation Language (3GL), there are some important limitations.

In the area of fast application development, 4GLs are impressive when generating routines of commonly used functions, such as simple file maintenance transactions. However, they are noticeably slower when non-standard functionality is required. For example, a user can spend weeks "bending" a 4GL product to produce more sophisticated functions, such as bar code routines. These routines would take only an hour to code in a conventional 3GL like COBOL. If applications require enhancements to meet corporate or market objectives, developing in a 4GL may not be the best alternative. 4GLs require that the programmer change the original specification and regenerate the code for the entire application. A 3GL allows you to make simple modifications to existing code.

A 4GL will also place additional overhead on the target execution system, because 4GLs are designed to select the closest fit option for code generation. The elements adaptable by the end user often lead to the production of complex reporting tasks that consume large amounts of CPU time. Additional hardware may be required to accommodate a newly acquired 4GL.

While 4GLs have an advantage in handling simple data manipulation, they do not allow the degree of customization, flexibility or ease of maintenance required for serious data processing operations on open systems.

6. Commercial Cooperative Processing

One significant trend in commercial application development and execution, is the every increasing instance of cooperative processing installations. In order to understand how DOS and UNIX COBOL development environments can play a significant role in the implementation of network based applications, it is important to first distinguish between cooperative processing, and client/server processing.

Cooperative processing is defined as two or more complementary programs interacting and executing concurrently on two or more machines as part of an overall business function, whereby each component exploits the characteristics of its respective platform.

Client/Server is a subset of cooperative processing in which a programmable workstation executes some portion of the application logic (beyond terminal emulation) including, but not limited to, the user interface.

A client is defined as the driving or initiating component of an intelligent workstation, delegating predefined types of tasks to a server and making requests of a server, usually for which it awaits a response. A server is a host acting on behalf of a client for a class of functions, e.g., database requests.

There are three main instances of client/server applications:

Remote presentation

Both data management and the application function are performed on the server. The application user interface is performed on the client. This is proving not to be a dominant architecture as it does not make good use of the power of the desktop client system.

Distributed function

Data management and part of the application function are performed on the server. The remainder of the application function and the application user interface are executed on the client.

Remote Data Management

Data management is performed on the server. The application function and the presentation of the application user interface are performed on the client. An example of this is a configuration where the database resides on the shared server, while the manipulation, report generation and presentation of the data are carried out on the client.

The creation of cooperative processing applications requires tools that embody the flexibility and performance of 3GLs with the visualization and modularity of 4GL forms generation packages. The objective is to off-load the more mundane IS programming tasks such as changing a report layout or generating custom reports to users by providing them with tools to generate their own reports.

The following section of this paper reviews a specific example of DOS, OS/2 and UNIX client/server distributed function, and the tools necessary to create the network application.

7. DOS and UNIX: Distributed Function

PCs executing DOS are the desktop systems of choice. They execute word-processors, spreadsheets and databases. They also act as gateways to central, corporate data repositories. Large UNIX server systems are ideal as data storage systems. UNIX is a mature, scalable operating system that enjoys widespread industry support and attractive price/performance.

A number of benefits can be realized from a network of PCs connected to a central UNIX system acting as a database server:

- The largely untapped power of the client PC is utilized to manipulate and present data downloaded from the server in the way that the user wants, not as defined by an IS organization.
- The server, which is running an interrupt driven operating system, is not wasting its time driving computer displays, it merely downloads the client requested data in the most optimal way and proceeds to handle other client requests.

For commercial applications, Micro Focus COBOL and Dialog System are the development tools that make the construction and maintenance of distributed function client/server applications simple. A COBOL application performs the data storage and retrieval on the server while Dialog System brings considerable power to the way that the human interface to the application is designed, implemented and maintained on the client.

Dialog System: The Tool

Before the role that Dialog System plays in a distributed function client/server application is presented, it is important to understand more about Dialog System.

The challenge that faces all developers of applications is how to best design, code and maintain a user friendly interface. All applications are comprised of code to perform the core processing task, data storage and manipulation, and a method in which the application presents information to users and gathers data from them. The traditional approach to the construction of application user interfaces has been to write large amounts of highly specific code to perform the data display and information input task. It has always been difficult for the developer to visualize how sequential lines of code will "look" when creating a visual screen display. Changes to the design or layout require complex source code changes and recompilation, consequently code is often repeated and complex.

Structured programming techniques teach us how to modularize applications. Application Program Interfaces, or API's, are clearly defined code interfaces that allow for structured, reusable code modules to be interfaced together. Structured Query Language (SQL) is an example of an API. SQL provides a clean interface between the process element of an application and a Relational Database Management System (RDBMS). The RDBMS is provided by a third party, removing the need for the application developer to design and implement an entire suite of data management programs.

Micro Focus has followed this model with the introduction of Dialog System. Dialog System is a user interface definition and execution tool that supports a simple API between the process element of an application and the human interface element. Using Dialog System, the developer can rapidly prototype the human interface to an application. The developer uses the Dialog System screen set definition tool to paint how the screen should look to the user. The screen layout design task is performed independently from the development of the process element of the application and can be updated and revised at any time. Dialog System supports powerful features to allow screen sets and the data that is to flow between the user and the application, to be rapidly defined. Screen sets can be enhanced or updated very rapidly without change to the application.

Dialog System dramatically reduces the amount of sequential COBOL source code that needs to be written by up to 75 percent. Because the COBOL application is smaller it is easier to understand and maintain. Maintenance of the application and the human interface become independent tasks.

What is Dialog System?

Dialog System comprises a development environment for screen definition and testing and a Run Time Environment for the deployment of completed applications.

The development environment includes a series of tools that are used to define the layout of the data to be passed between the application and Dialog System. Painting tools are used to define overlapping "panels" of information and how these panels interact. There is also a test execution and debugging tool that allows the developer to fully test a Dialog System screen set completely independently of the application.

The Run Time Environment supports screen sets that are called using the standard COBOL CALL verb. The Run Time Environment manages the exchange of data and control information with the application.

Dialog System -- Distributed Function Application Model

It can be seen that because of the clean API, the CALL interface that controls the flow of information between the application and Dialog System, there is no constraint that the application must be executing on the same hardware system as the Dialog System Screen Set.

A remote PC client can support the Dialog System Run Time executing a Screen Set. The link between Dialog System and the application is maintained between the client system and the server system, where the application is resident, by a communications package. A single server is capable of supporting many client systems running different Dialog System Screen Sets. Different Dialog System Screen Sets can be used, all supported by the same, single application without change, and all taking maximum advantage of the display technology and intelligence provided by the client. Complex, color based Screen Sets can be executed on DOS, OS/2 or UNIX Workstations connected to the remote server. This is because the intelligence and power needed to control the display is local and available.

Simple Dialog System Screen Sets can be executed on low cost, serial terminals supported by the server itself. Because UNIX is an interrupt driven operating system, it is important to minimize the amount of operations performed on serial terminals as they rely on the UNIX processor to service each and every keystroke. This is in stark opposition to intelligent, client workstations that only interrupt the UNIX server when data, input by the user, has been locally validated and has been sent via the communications package to the server for processing. The application has no knowledge of the display technology being used in any instance.

8. Application Portability

When discussing the benefits of commercial application development for DOS, OS/2 and UNIX, it is important to consider the needs of non-corporate users such as Independent Software Vendors (ISVs). The ability to run a single version of an application on different hardware platforms supporting different operating systems is one of the prime open systems goals from which ISVs can benefit.

The C and FORTRAN path towards application portability is to write applications in a way that is hardware-independent, using standard versions of the programming languages. However, because applications are distributed in a compiled, "object code" form, and object code is specific to a particular hardware architecture, users must purchase and install a different version of each application they want to use for each different computer architecture they want to run it on. ISVs must, at minimum, modify, recompile, relink, repackage, and redistribute their applications for each hardware platform that they wish to make their application available for.

For ISVs, the inability to capitalize on the market potential of their products is due, in part, to the need to divert resources from product development into product porting and maintenance. For hardware vendors, the result is the limited availability of applications for new, innovative systems and, consequently, a higher risk associated with innovation.

Micro Focus COBOL achieves the application portability vision for open systems. The Micro Focus COBOL compiler enables developers to compile and distribute their applications in a form that can be installed and executed on any hardware architecture and operating system combination that supports a Micro Focus Run Time Environment (RTE). Micro Focus compiler intermediate code is hardware and operating system independent, requiring only the libraries that constitute the Micro Focus Run Time Environment to allow the COBOL application to be fully executable.

This brings a number of benefits to the end user:

- Increased availability of software for open systems.
- Ability to decouple software and hardware purchasing decisions. End users are able to buy hardware systems with the knowledge that their [COBOL] application will execute on whatever hardware system they chose.
- Ease of distribution throughout large corporations, across multiple hardware architectures.
- Reduction in training costs when end-users move from one platform to another. With the availability of the same applications on a variety of platforms, the transition for end users from an obsolete to a new platform will be less expensive.
- Increased longevity of software investments because of the de-coupling of software from hardware obsolescence. COBOL applications will run on new hardware that support the Micro Focus Run Time Environment.
- Scalability of applications. The same COBOL application will run on all types of systems -- from microcomputers to super computers.
- Reduced need to upgrade an application for new system versions. When new versions of an operating system are released along with new releases of the COBOL RTE, the original COBOL application will execute unchanged.

Application developers benefit from cost reductions in software development, maintenance, testing and distribution when they write their application in machine independent COBOL. Other benefits include:

- Simplified application development and increased product consistency across platforms.
- Reduced maintenance costs by limiting development and support to a single source version of an application.
- Reduced testing costs.
- Reduced manufacturing and distribution cost as only one version of an application is produced.

9. Conclusion

In 1991, single user DOS and OS/2 systems constituted half of the $186 billion worldwide systems market [2]. 1.17 million UNIX systems were delivered in 1991 representing $16.5 billion in revenue. The fastest growing segment of the UNIX market is workstations, representing 51.1 percent of all UNIX revenues. It is generally believed that UNIX will achieve no more than a 10 percent share of the commercial desktop market; but will have a large share of the server market. [3]

Traditional computing has focused on optimizing the application for the most efficient use of the mainframe or minicomputer. PC and workstation development emphasize optimization based on the users needs, even when this requires more code and greater execution system power. The PC approach provides dividends in terms of greater user productivity, better decisions and lower end-user training, support and operations costs. When coupled with the maturity, power and flexibility of the COBOL language and COBOL application development tools, the developer becomes more productive, and system and network resources are used more effectively.

References:

[1] InfoCorp. *Trends in Software and Systems*. 1991
[2] IDC.
[3] Gartner Group.

CIB
PC:\lib\papers\trend.doc

Computational Clusters from Hewlett–Packard Company

Background

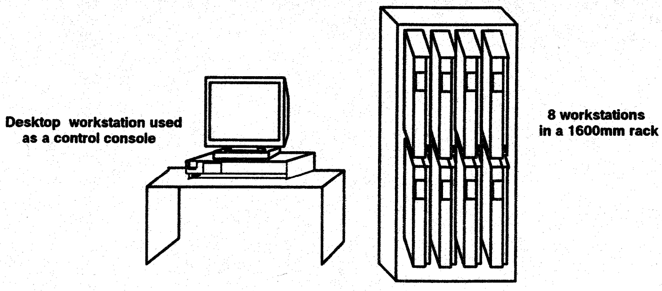
Cluster computing has become popular as a result of two trends, one economic and one technical. On the economic side, many mainframe and supercomputers have been leased and the leases are ending. Even supercomputers that have been paid for cost approximately a million dollars a year to keep running. As a result, many high performance computing users are evaluating solutions other than traditional supercomputers to run their numerically intensive computations.

On the technical side, RISC workstations have continually improved in performance at a very fast rate since their introduction in the mid–1980s. RISC processor performance leap–frogs nearly every 18 months; supercomputer performance continues to improve also, but not nearly as fast. Another important fact is that, while workstations are getting faster, they are also becoming more affordable. Supercomputers, on the other hand, are becoming more expensive with time.

Today, many scalar applications used in supercomputing (such as the commonly used Monte Carlo simulation) actually run faster on individual RISC workstations than they do on the fastest supercomputers, since traditional supercomputers rely on specialized vector processors for most of their speed. In fact, many users have successfully utilized the power of multiple workstations to solve their compute–intensive tasks. There is much work underway to improve the hardware and software technologies used in workstation clusters.

Computational Cluster Definition

A computational cluster is a group of two or more workstations networked together and used as a virtual single computational resource. Workstation clusters can augment or replace the computing power of traditional mainframes and supercomputers to perform batch or parallel processing. As shown below, a computational cluster is configured with two or more headless workstations connected via Ethernet or a higher-speed, lower-latency networking link such as FDDI, HiPPI, or, in the future, FibreChannel and ATM (Asynchronous Transfer Mode). A workstation or server connected to the cluster serves as a console to manage the workstations in the cluster. Software residing on the front end machine manages the task distribution.


Desktop workstation used as a control console
8 workstations in a 1600mm rack

Example Workstation Cluster

Cluster Advantages

Workstation clusters are, by their nature, modular and flexible. In the past, users would actually buy more supercomputing power than they needed at the time to have room to grow over time. Now, with workstation clusters, users can simply add more workstations to the cluster as their needs grow or swap out older systems with more powerful systems. Additionally, users have flexibility in configuring clusters to meet this specific application and environmental requirements.

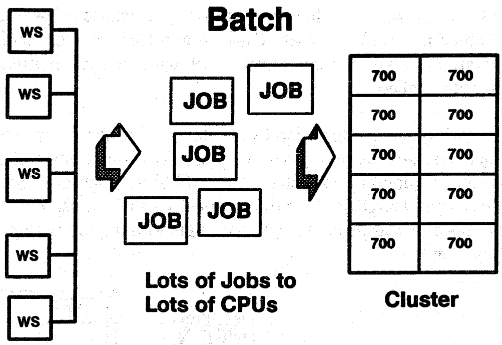
Many researchers are also looking at clusters as a low–risk entry into the world of Massively Parallel Processing (MPP). Many believe that an industry standard will develop with MPP machines and are afraid of locking into the "wrong" technology. Until a standard emerges, many researchers are using the workstation cluster as a platform to develop their parallel programs. The binary compatibility between workstation clusters from Hewlett–Packard and the SPP from Convex will make it possible for scientists to run parallelized applications developed on a cluster on a MPP machine.

Cluster Scalability

The size of a cluster may vary according to the application and the processing mode (batch or parallel). The maximum number of workstations used in a batch processing cluster is unlimited, with current installations topping 100 workstations. The optimal number of workstations in a parallel processing cluster ranges from 8 to 16. Although, the optimal number of nodes in a cluster is largely application dependent. Beyond the optimal number of workstations (for a specific parallelized application), the message passing between machines starts to overcome the additional useful work being done.

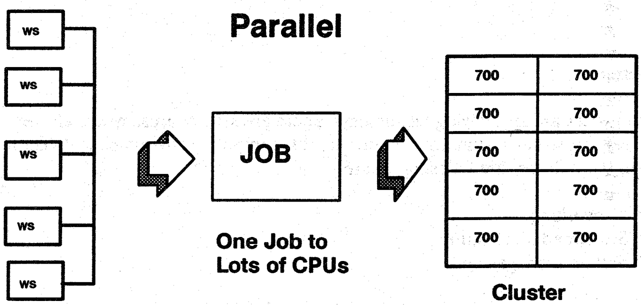
Batch Processing with Clusters

Here a computational cluster acts as a throughput accelerator. This mode is the most popular way of using computational clusters because there is no impact on application code. As shown in the following diagram, when a computational cluster is in batch processing mode, jobs are sent to the cluster from workstations, X terminals, and other available clients. Tasks are automatically sent to the cluster via a central queue or a distributed queue, depending on the batch queueing solution used. Jobs sent to a queue are large compute–intensive calculations, background jobs, and other non–interactive tasks which typically require more than ten minutes processing time. Batch processing is useful for applications such as CAE, CAD, software development, and scientific analysis. For example, a software developer uses batch queueing software to run compiles on a cluster while the developer's own workstation is free to do more interactive development. The batch software, such as Task Broker or NQS, performs functions such as task distribution and balancing the workload to maximize performance.


Batch
WS
WS
WS
WS
WS
JOB
JOB
JOB
JOB
JOB
Lots of Jobs to
Lots of CPUs
700
700
700
700
700
700
700
700
700
700
Cluster

Parallel Processing with Clusters

When a cluster operates in parallel processing mode, as shown in the following figure, a large job sent to the cluster is divided among multiple CPUs. Parallel processing is an advanced form of clustering and requires modifications to the application software to break it up into relatively independent tasks. The customer uses software tools such as Linda or PVM to help parse the program into smaller pieces of code or information. In a parallel processing cluster, interaction between processors can be slower than in a traditional multiprocessor computer. Applications must have a high ratio of compute-intensive calculations to I/O communications to perform well in a cluster computing environment. This type of application is called a *coarse-grained* application. Molecular dynamics, ray-tracing used in visualization, and seismic data analysis are examples of coarse-grained applications.

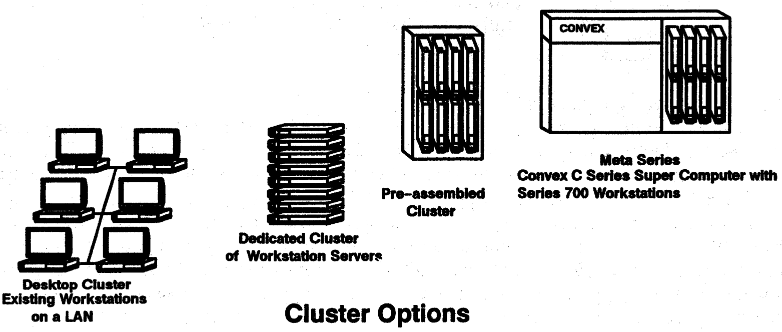

Parallel
WS
WS
WS
WS
WS
JOB
One Job to
Lots of CPUs
700
700
700
700
700
700
700
700
700
700
Cluster

The Ways to Cluster

There are several ways to configure a cluster. The simplest way is to create a "**desktop**" or "virtual" cluster composed on the workstations on the LAN. With the addition of the right batch or parallel processing software, existing desktop workstations are utilized as a single computational resource in off-hours to do batch or parallel processing in addition to the interactive design work they were purchased for.

Many people have already set up "**dedicated**" clusters of segregated workstations in addition to those on the desktop to off-load the very computationally intensive parts of their work to free up the interactivity of the desktop client. Typically several high-performance workstation servers are set up for this task. Additionally, customers now have the option to purchase a "**pre-assembled**" cluster, which is already assembled in a rack. Hewlett-Packard and Convex offer configurations of up to eight Series 700 workstations can be mounted into a rack with the system already pre-configured with the selected network installed.

The highest level of clustering is to create a MetaComputer, a computer composed of different types of processors. The object is match the right job to the right processor. Convex offers a "**MetaSeries**" containing a C series supercomputer and a cluster of 2-8 HP workstations. The Meta Series integrates the vector processing and large data handling capabilities of a supercomputer with the scalar processing of workstations and is appropriate for customers who require both processing methods. This product is sold by Convex.



Desktop Cluster
Existing Workstations
on a LAN

Dedicated Cluster
of Workstation Servers

Pre-assembled
Cluster

Meta Series
Convex C Series Super Computer with
Series 700 Workstations

Cluster Options

Future Directions

Workstation clusters are an evolving technology. There are several areas where cluster technology can be improved to provide a more robust and versatile computing environment. Specific technologies leaders in the industry are focusing on are

* Networking
* System administration
* System management

Areas of investigation in networking include higher bandwidth, lower latency connections between machines in a cluster. In addition to FDDI, networking products under development include HiPPI (High Performance Parallel Interface), Fibre Channel and ATM (Asynchronous Transfer Mode). With faster interconnects, more finer grained parallel applications may be ported to a cluster. Additionally, we are studying methods to reduce the protocol stack (used for message passing) to further improve communications between processors.

In addition, improving the methods with which to administer and manage a workstation cluster will make a workstation cluster appear as a single system.

#3010

Evaluating X On The PC

Matt Hulett
Walker Richer & Quinn, Inc.
2815 Eastlake Ave. East
Seattle, WA 98102
206-324-0407

Introduction

The X Window System has evolved significantly from its origins at MIT. Today, X is the most widely supported solution for distributed computing from the desktop. The chief attraction of X is that it is an open standard providing true interoperability. While X is still used more frequently in technical applications, shrink-wrapped and in-house applications are being developed. This phenomenon is accelerating the use of X in corporate computing.

From the desktop, X is primarily run on X terminals, workstations, or PCs. There are specific instances when each of these platforms is optimal. Typically, workstations and X terminals are used for performing graphic intensive applications (i.e., CAD/CAM, GIS, etc.). The PC is generally used to access X applications that have traditionally been in the domain of MIS (e.g., databases, electronic mail).

Any organization that has implemented, or is planning to implement X, needs to consider PC-X integration. PC X servers often provide a practical connectivity solution in organizations with a large installed base of PCs, but it s important determine what platform makes sense in each computing environment.

Components of X

X can be best described as a hardware-and network-independent windowing and graphics protocol. It is a software standard providing a framework for window-managed applications to run over a network.

In the X Window environment, end-user applications are called X clients. The client is the computational portion of an application residing on a host. The X server is the portion of the X environment that handles the front-end display tasks. The X server software can run on a variety of machines—PC, Macintosh, X terminal, UNIX workstation, etc. The server's purpose is to control the display and input devices such as the keyboard and mouse. It performs the fairly passive role of listening for commands from clients and input from users (key presses and mouse clicks), which it passes on to the client.

To display objects on the screen, the program makes calls to various X run-time libraries. The highest level of object functionality is provided by a library called a widget set or toolkit. Widget sets implement facilities like pull-down menus and text windows. The most important widget sets are the Xm Motif toolkit, and the Xol OpenLook toolkit. At a lower level come the Intrinsic functions, which provide basic drawing and display capabilities; almost all X Window programs use the X Consortium's Xtk intrinsics. The lowest-level interface is Xlib, a subroutine library that performs the actual interaction with the server over the network connection.

Another important component is the concept of a window manager. A window manager is a program that controls windows on the screen. These programs control different windowing policies like minimizing and maximizing, scaling, and the title bar. In X terms, the window manager is a special X client which can be run (either locally or remotely) for each X server. This program is given special privileges and is allowed to "supervise" all of the windows being displayed by the X server. The window manager will typically place some form of window decoration around the outside of each X client window that includes resize and move buttons as well as a title bar. It then becomes a function of the window manager to resize, move, or rearrange a window according to the wishes of the user by mouse clicks on a window's decoration or selections from a window manager menu.

At present, there are several managers for the X Window System, the most prominent of which are the OSF/Motif, Open Look and the Tab Window Managers. Also, kernel-based window managers like Microsoft's MS-Windows or the Macintosh. It is important to note that the window manager only creates the "look and feel" of an X client with its window decoration. Whatever an X client chooses to display in its output window is independent of the window manager. Program libraries are available to X Client developers that allow them to create an application with a specific look — either an Open Look or OSF/Motif look, for example. As mentioned above, these tasks are handled via toolkits.

X Advantages

There are several advantages to the technology. X provides significant productivity for users. Since all X applications are written to support a single display device, the X Window System allows users to execute applications running on different machines across a network. X provides a common interface across multiple platforms and X shields users from different systems. This results in applications having the same "look and feel" across any desktop platform, reducing training time.

X provides advantages to developer's, too. X allows a developer to write a single program for use on any host. Since X splits an application into two functional halves, the developer only needs to write the client portion. The front-end portion is already specified within X. Perhaps the most attractive advantage of the X Window System is its solution to interoperability. X allows for a distributed computing scheme: Applications can run on any machine and can be displayed on any machine. And since it is not specific to any one vendor, X is very attractive to organizations in a heterogeneous computing environment.

Users Of X

The X market has been traditionally technically-based, although, X has been making increasing gains in the commercial sector. X use is popular in industries like financial institutions and insurance companies. Other industry sectors that heavily use X are aerospace/defense, manufacturing, petro-chemical, business services, education, health care, banking and finance, telecommunications, retail distribution, as well as government agencies.

For instance, X is very popular in organizations that have commercial transaction-intensive telecommunications operations (such as the regional bell operating companies) and financial service environments (such as American Express), where customer and billing information may reside in separate applications on different hosts. X, by its very nature, can tie the computing network together, providing a common interface for different platforms on a single.

X Application Environment

As mentioned earlier, X has technical roots, which has influenced the areas of application development. Users of UNIX workstations are technical users such as engineers. Typical applications include CAD/CAM, statistical modeling, etc. The move of X into the commercial sector has attracted some major application developers like WordPerfect Frame Technology, Lotus, Oracle, Informix. But, X is still in-house technology. Corporations are building custom database front-ends and customer service systems. In other words, X is being used for mission-critical applications.

The majority of X sites are UNIX-based. X is the standard windowing architecture for UNIX. Since the UNIX market is growing the X market will grow ,too. One of the primary complaints users have had about host-based systems is the cryptic nature of their user interfaces. In particular, the UNIX command line has been a prime target. X addresses that problem by letting developers build friendly interfaces for UNIX-based legacy systems, making UNIX-based applications accessible to non-technical users.

The development of these custom applications is easier with the growing popularity of GUI builders. These types of applications are useful since X toolkits provide low-level functionality. Coding X applications was, at first, an excruciating process of tinkering with display parameters, positioning, and sizing, often through editing pixel values in C source code. GUI tools are used for building GUI programs which allow the developer to edit the application's appearance graphically.

Which Platform to Choose?

There are three main platforms for X Window implementations from the desktop: X terminals, UNIX Workstations, and PCs. The X Business Group estimates that in 1992 there were approximately 2.1 million X seats worldwide. Of these, 60% were UNIX workstations, 19% were X terminals, and 21% were PCs.

The UNIX workstation segment is an easily defined customer base made up of highly technical users. These users typically require graphically intensive applications, such as CAD/CAM, as well as substantial local processing power. The fact that workstation users make up the majority of X use is also a strong indicator that X is heavily used in the scientific and technical sectors.

It is becoming increasingly difficult to determine whether it is better to implement a PC running an X server. Because an X terminal is a dedicated X server, performance levels are naturally better than those of a PC. There are also some environments where X terminals are the preferred implementation. For instance, sites with efficient network resources, centralized management, and a need for a very high level of security often require dedicated X devices. Not every data processing department wants to exchange its dumb terminals for intelligent workstations. Low cost, diskless terminals are more secure than workstations and PCs with local storage; they run only those applications you want users to access. For such departments, dedicated terminals using the X Window system provide an inexpensive way to run graphical applications on a network.

The PC Meets X

Recently, PCs have become a more popular platform than X terminals. As PCs achieved more network capabilities and their relative performance levels have risen, they are handling more difficult computing tasks. Also, the stability of PC X servers has improved, making them much more attractive.

In addition to the advances in PC technologies, PC X servers have an inherent advantage over other desktop platforms. The main reason organizations choose PC X servers is to leverage the investment in their PC installed base. Users who once only needed access to word processing, spreadsheets, and other locally-based applications now often need to work on mission-critical X applications too. Instead of adding another device to the desktop, such as an X terminal, a PC X server package can be added as a low-cost solution.

The savings on hardware costs is not the only benefit. PC users are usually reluctant to add another computing device to their desktops. The typical PC user is very comfortable with his or her current environment and does not want to learn how to use another platform. The chief advantage to users is that putting an X server on the PC lets them easily integrate two environments. An MS-Windows-based PC X server lets the user cut a graphic from an X-based drawing package and paste it into another Windows application like MS-Word for Windows, which saves a lot of time and effort.

PC X server functionality

Options for implementing X on a PC are OS/2, DOS, Macintosh, NT, and Windows. Since, the most popular platform is a Microsoft Windows-based PC, let's focus on common features found in Windows-based PC X servers.

Vendors of PC X servers typically offer a basic feature set. For example, most PC X servers support X11R5 (X Window System version 11 release 5) that was released in September of 1991. At the very least, a PC X server should support all of the major X Window GUI standards, including Motif, Open Look, DECwindows, and others.

Support for multiple-and single-window modes is usually provided, too. Offering different window modes lets users choose either the local window manager in Microsoft Windows or a remote X Window manager, such as Motif or Open Look. Remote window managers generate significant network traffic, because window management functions (window resizing, minimizing, iconifying, etc.) are remote processes. With Microsoft Windows as the window manager, network traffic is reduced, since the windowing operations can be performed locally. Also, by using the local window manager, users benefit by having the same Microsoft "look and feel" with their X applications.

Microsoft Windows-based X servers also provide the ability to cut and paste data from one application to another. PC X servers make sharing information between X and Microsoft Windows virtually seamless, which can make a real difference in productivity. For example, a user could paste chart from an X application on an HP 9000 and include it in a word processor document on the PC.

More important are the issues of performance and reliability. Running two windowing systems — X and MS-Windows — is very demanding for a PC. The minimum requirements for a PC are at least a 386 with more than 2Mb of RAM. Of course, the more powerful the PC the better. Reliability is an issue since X is used with key applications. PC X servers should be tested in-house and evaluated with a company's home-grown applications.

PC X servers should offer a wide variety of support for networking software. Since their are many vendors that offer TCP/IP implementations. It is important to find a vendor that effectively supports the appropriate stack.

Conclusion

As companies increasingly migrate to UNIX to run their mission-critical applications the need for smooth integration between the PC and UNIX is essential. The decision concerning what platform to implement X on will become less distinct as PCs become increasingly powerful. In fact, the as the lines between PC and workstation performance start to blur, the PC will become a mission-critical machine. In respect to X, each organization should consider their particular needs and how the various desktop platforms provide a solution. But, consideration should be given to the undeniable trends within the PC industry and Microsoft Windows, for any organization looking to the future.

ABSTRACT

LIVING WITH OPEN SYSTEMS

HOW TO SURVIVE ACRONYM SHOCK

Speaker:

Rikki Kirzner
Senior Industry Analyst
Dataquest
1290 Ridder Park Drive
San Jose CA 95131
(408) 437-8325

Abstract

There is a tremendous amount of fanfare from vendors about standards. However, the end user is faced with the dilemma of sorting the wheat from the chaff (or the DCE from the DME). In addition, vendors sometimes specify complete standards when only a subset may apply to the product or the needs of an organization. This paper will cover what standards are important, how end users can benefit from standards, and what it means when a vendor says they are "compliant with IEEE POSIX, XPG3 and XPG4, and the SVID".

Standards, plus their associated committees, organizations, and consortiums have been created to either clear up or confuse the issue. A few that will be discussed include:

IEEE
ISO
OMG
SVR4
TCP/IP
UI
WINDOWS
DME
FIPS
DCE
OSF
MOSES
CMIS
POSIX
SVID
GOSIP
XPG4
CORBA
MOTIF
CMIP
CAIRO
ODBC
WOSA
COSE
88OPEN
OPENLOOK
SNMP
ABI

Some of the questions that will be answered include:

Why do we need standards?
What is meant by "portability", "interoperability", "source portability", "binary portability", and "compliance"?.
Which standards are important to my organization, and why?
How "open" are Open Systems? What does "open" mean?
When does a proprietary system become an open system?
When does an open system become a proprietary system?

#4002

The Special Requirements of Commercial UNIX Data Centers

Tom Harris
Operations Control Systems
560 San Antonio Road, Suite 106
Palo Alto, California 94306
(415) 493-4122

OVERVIEW

While any major transition from one operating system to another is vulnerable to slipped schedules and cost overruns, the transition from a solely MPE-based platform to the mix of MPE and UNIX holds unique surprises. This paper discusses problems related to system administration, so management can anticipate them, and adopt a proactive stance to provide a more cost-effective and better planned transition.

PROBLEMS ADDRESSED

The conversion from a strictly MPE environment to a mixed MPE and UNIX environment presents some new challenges because of the radically different philosophical bases of each operating system. In addition, moving from the comfort and predictability of MPE to the unknown UNIX provides a variety of new technical delights, but it can also open a Pandora's box full of security and audit problems. While new state-of-the-art technology, attractive and easy-to-use graphical user interfaces, and robust networking become possible, the shock of a comparatively hostile operating system interface combined with accountability problems produces significant management challenges. This paper will present installation options for using the standard MPE and UNIX utilities and will offer suggestions for supplementing them with better software so that your shop will benefit from the decision to bring in UNIX.

System administration includes hardware and software configuration as well as ongoing maintenance tasks to provide data processing facilities to the user community. The administrator performs many functions in order to manage the multiple resources in the UNIX environment. These resources include CPU and peripheral hardware, user accounts, operating systems, and utility programs. Please note that we use the plural since most shops have more than one UNIX system. The administrator is faced with installing network hardware and software systems. Often, the computers are not of the same type, and a management nightmare is born.

Since the management tasks are many, the administrator needs programs and procedures that are easy to set up (install and configure), time efficient, easy to use, offer good performance, and result in high system availability. We also know that the administrator needs reliability so that tasks are repeatable and auditable by a third party. The MPE user is accustomed to getting hard copy audit trails of all system administration tasks. With UNIX, audit records are not always available.

SYSTEM ACCOUNTING

Management wants to know who is using what resources, how often, and when. In order to obtain this kind of information, a monitor process runs continuously in order to collect raw data such as user logins, programs executed, disk consumption, printer activity, and port connections. Once collected, the raw data is analyzed and organized into categories for management reports, user bills, graphical presentations, or data files. Very often, the summary information is exported to a master location and used for a corporate accounting application.

How is this done in an installation with both MPE and UNIX systems? With your MPE eyes, you see that system logging is accomplished by the monitor process. The MPE logfiles hold the raw data, some standard utilities (LOGUTIL) and command (REPORT) functions organize the data and generate reports for the system administrator. Third-party software takes this reporting to a higher level with price tables, consolidation options, discounts, minimum charges, and multiple-machine presentations.

Your UNIX system also collects system accounting information. The kernel monitors all tasks, the *wtmp* and *pacct* files store the raw data, and a suite of commands (*acctcon1*, *acctcon2*, *dskusg*, *runacct*, and many, many others) organize and report system accounting information. Only two price schedules (prime time and non-prime time) are standard with UNIX and few statistical reports are available.

The problem here is complexity. Your management wants more information for accounting and security purposes. First of all, we all want to know who is printing what, when, and how much. Then we want to have a complete picture of the installation. This means having system accounting utilities that capture and combine information for all machines in the organization. And with today's sophisticated graphical presentation tools, why settle for the same old reports when you can get a picture?

With your MPE eyes, you expect to see pricing tables, rate schedules, quantity discounts, minimum charges, and much more in the UNIX environment. This should be the minimum requirement.

WORK LOAD MANAGEMENT

Imagine you have 1700 UNIX boxes in separate locations all over North America. Your organization has an Order Processing application in place where some users enter orders, others send the orders throughout the network and, at some time during the day, the orders are summarized and acknowledged in a warehouse. This open systems environment has a client/server architecture and there is a great deal of work that must be done by the user community.

The system administrator cannot have one person at each of 1700 stations throughout the nation. He can use work load management software to schedule tasks, monitor activity, verify task completion and, if necessary, recover from unsuccessful operations. We expect this software will look for optimization opportunities as it monitors the realtime operation. What we hope for is an efficient software product that works in the background and maintains audit trails of data processing activity.

What the system administrator wants is a program or set of programs that automate launching jobs or tasks, optimize processing for maximum throughput, and record activities throughout the network. To get started, the system administrator must establish rules for each location including such items as task, date, requirements, and recovery.

With MPE, the user establishes rules with the STREAM facility. The environment is further refined and tuned with job fences, priorities, queues, and job limits. The operating system dispatches jobs and the user can monitor activity with the SHOWJOB command. System logging generates an audit trail and users can perform manual STDLIST analysis to determine task completion. Automated recovery is not provided.

With UNIX, the system administrator uses an editor to define up to 26 processing queues and to set their characteristics. Users then establish their own job schedules with *crontab* for repetitive tasks, and *at* or *batch* for one time requests. The *cron* program runs continuously in the background and dispatches work, monitors progress, checks the exit status, and sends the task status and output to the task submitter. The user can monitor activity status with the *ps* command (process status). For audit purposes, the *pacct* (process accounting file) is updated whenever a task is complete.

Is this what you need in a multiple system environment? With your MPE eyes, do you see enough automation to get the job done? You need system administration utilities to automatically manage the workload and maximize throughput. You need a big picture view so you can see what is scheduled, what is running and what is completed on all systems in your network. Equally important is the need to have an easy method for defining workload and establishing processing rules. In fact, more rule granularity is required than standard UNIX supports. Scheduling rules include task, time, date, and prerequisites such as previous task checking, operator prompting, and output content analysis.

CODE MANAGEMENT

Operators and programmers often fight over the issue of code management. Who is responsible for it? The system administrator is usually pretty clear about two functional areas, program development and file distribution. Engineering is usually responsible for the former, Operations the latter. With a dozen UNIX boxes for engineering workstations, we have a problem that needs a solution. These engineers are all working on the same project, but each has his or her own assignments. During the program development phase, the engineers need some basic checkout/checkin procedures to avoid getting in each other's way. When the development is complete, the operators need an automated move-to-production process to handle file distribution.

System administrators solve the code management problems in several phases. Engineers identify library files and set up an official master index. Then, system administrators locate critical files, identify user classes, and restrict access to specific users. Next, they establish a process for basic checkout and checkin so that the engineers' work is preserved and the software asset is protected.

If the code management issues are related to production activities, then the system administrator defines an approval process and file distribution technique to take tested code from the engineers and move it to a production location. Standard separation of duties are implemented so that untested code cannot get in to the live production world.

Code management is enforced with proper implementation and audit trails are automatically generated for tracing purposes. With any code management implementation, release control and version control are necessary. In the production environment, automatic recovery and code rollback are vital.

What do we have for the MPE environment? Nothing comes standard with the operating system, but there are third-party products available that handle checkout/checkin and sophisticated move-to-production procedures.

In the UNIX world, engineers have two standard packages, SCCS and RCS, for source code management (notice the word "source"). With SCCS, the *admin* function is used to set up a library. Engineers access files with the *get* and *delta* commands. Audit trails are maintained with the *prs* function. If the engineers prefer RCS, the *rcs* command is used to define the master library. Checkout and checkin functions are executed with the *co* and *ci* commands. The audit trail is maintained with the *rlog* command.

SCCS and RCS are entrenched in the UNIX world. But, with our MPE eyes, we demand a more powerful, comprehensive, multi-system solution. First of all, the administrative setup must be easy to use. We must be able to associate more than one file to a class category and then use that category to access a collection of related files. Enforcing procedures must be transparent to the engineer because, as we all know, they have better things to do than spend time learning new commands in order to gain access to their source code (and usually resent the access controls in the first place!). In addition, generating audit trails must be automatic and comprehensive.

Operations management needs an automatic method to distribute files across the network. Automation ensures that object code is matched to source code and that the tested version of object code is sent to the correct production platform. After all the hard work that engineers accomplish, it would be criminal to compromise on sloppy (or non-existent) procedures for moving code into a live environment.

CONCLUSION

Recognizing the significant differences between MPE and UNIX is a good first step to planning the System Administrative procedures for your environment. The demand for robust and efficient system utilities is obvious to the new user since the standard programs do not handle multiple machines and do not automate repetitive tasks. The utilities that do exist are difficult to use and have limited online help facilities. When the EDP audit is performed, the administrator will have a hard time finding xcomprehensive audit trails and evidence that disaster recovery procedures are available.

Don't settle for less than you need! You will pay for it twice – once when you identify the missing utility, backtrack, and locate whatever evidence may exist, and a second time when you write your own comprehensive utility program. How much pain are you and your users willing to endure? Who do you want to pay? How much are you willing to pay? When do you want to pay?

Be strong and take action before this environment eats you and your users alive. It is not smart to grin and bear it; you are advised to either write your own system administrative utilities or purchase third-party software. The decision is yours after you estimate the costs of make versus buy. Remember, you deserve the best!

PAPER NUMBER: 4003

TITLE: Getting Your Money's Worth Out of Network Computing

PRESENTER: Dave Mahler
Remedy Corporation

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

DELIVERING MULTIMEDIA IN A NETWORKED ENVIRONMENT

William L. Kemper
Open Systems Software Division,
Hewlett-Packard Company
1000 N.E. Circle Blvd.,
Corvallis, Oregon 97330 USA

ABSTRACT

Multimedia technologies should expand an individual's communications potential and increase the effectiveness of communication in a networked environment. HP MPower is an integrated multimedia-enriched graphical user environment that is delivered and installed in a client and server configuration. This paper will describe the capabilities of MPower and the delivery mechanism for the product in a networked workstation environment.

Multimedia Technologies: Concepts

Multimedia technologies have been delivered on both personal computers and workstations for several years, but they have not been embraced as mainstream technologies. The reasoning for this is twofold: a lack of general purpose application and costs. While the creation of slide presentations, audio tapes, and video clips, all have relevance in the workplace, many applications utilizing multimedia information are designed with the professional graphic artist or video technician in mind. These applications do have their merits, but the expertise required for competent usage is beyond the casual user's level. The computer has provided a powerful tool for all users, but has been limited to simple data, i.e. text and numbers. Pictures and audio are readily usable bits of information, but they have been left out of the typical data form on computers. These data types require additional hardware and software in order to fully utilize the information. In the early 1980's, several people were grappling with the personal computer. Here was a device that could provide computing power on one's desk. The first applications were simply adaptations of applications that had been written on mainframes and minicomputers. The paradigm was simple. The user entered data, the computer crunched, the answers were given usually displayed or printed. The concept of an electronic spreadsheet was unheard of and the personal computer was limited to a small set of hobbyists and engineers. VisiCalc and 1-2-3 revolutionized the usage of the personal computer. The user still enters data, but gets feedback interactively. Information can be analyzed, graphed, printed, and understood. The application has made the computer an essential tool

Multimedia As Information

Multimedia is information. People make decisions, take action, understand others with the communication of information. The electronic spreadsheet gave people the power to communicate information in a usable and dynamic form. Images, audio, and video can be powerful sets of information. Electronic mail has brought people closer, but has been mostly used for messaging. The evolution of communication from the telegraph to the telephone has increased communication by increasing the information content and removed Morse code as an interface barrier. Treating multimedia as extensions of information types will improve the communication of ideas and the depth of the information.

Building Upon Existing Technology

Delivering a fully enabled multimedia environment needs to utilize existing technologies as much as possible. Voicemail systems have had tremendous growth over the past few years. I contend that this growth is directly tied to the ability of this technology to leverage existing technologies and user paradigms. The system could be installed in a generally non-invasive manner. Most users of the voicemail system are unaware of the technologies that are being utilized. The phone, a very familiar device, has a few new functions that make use of the existing hardware. For some people, voicemail may have been the first time that they even used two of the buttons on the phone,(* and #). The concept of leaving a message on a machine was readily understood. Answering machines were just tape recorders tied to the phone. Voicemail took advantage of these basic concepts. Which leads to workstations, the UNIX operating system, and multimedia hardware availability. Workstations are networked devices. Data is readily transmitted across the network. FAX machines are everywhere, even in people's homes. Audio and image data are simple extensions of text which is readily utilized in everyday communication. Image and video hardware have been developed that can bring this information into the computer.

The Workstation Solution

By focusing on the basic task of communication, multimedia is a logical extension of existing forms of information, e.g. text, data bases, and slides. The workstation has the computational horsepower to handle this information explosion and can deliver it across the network. Many disparate pieces exist, but have not been integrated into a consistent environment. The user has been given several tools, but not empowered with a cohesive environment. Returning to the spreadsheet analogy may shed light on the situation. Separate applications existed to perform curve fits on data, graph results, print tables, but the spreadsheet brought these separate applications into a single application. The spreadsheet has become a tool that can be customized to specific applications, e.g. a pricing analysis, an engineering analysis, etc.

HP has integrated the user environment with multimedia technologies in the MPower product offering. This solution is designed for workstations and recognizes the networked nature of these machines and the UNIX operating system.

The User Environment

The development of the graphical user environment extends the interaction model of the computer. A simple point and click or drag and drop allows users to readily interact with applications in a more intuitive manner. By standardizing the user interfaces, people can depend on consistent behavior. Basing development on MIT's X Window System and the Motif environment delivers a consistent behavior. Human factors can be utilized to improve usability. Usability testing can provide immediate insight into design flaws. Visual cues, while seeming elementary, can significantly improve the immediate understanding of the user interface.

User Perspective

In order for multimedia to be a readily accepted technology, barriers to acceptance must be identified and eliminated. Some of the obvious barriers are the invasive nature of audio and the lack of integration of the FAX machine into the computer environment. While some of the barriers could have been eliminated in the past, the costs outweighed the perceived value of the information. The solution needs to be scalable from a basic level to a more robust, albeit costly, set of technology. The basic level needs to deliver an immediate return to the user that extends their communication capabilities. The user paradigm needs to be a simple extension of the existing data model. In MPower, audio and images are just new data objects that can be exchanged in the same manner as a paragraph of formatted text. If the recipient has an application that can decode the formatting specification, the full content of the object can be seen in the manner in which it was created. If the application does not have the decoding capability, the object must be identifiable so that proper action can be taken.

Online Help

No matter how much developers would like to think that their solutions are readily comprehensible by any user, there is a need for documentation. Most manual writers know that their work is generally ignored unless the user has reached a level of frustration that verges on the violent before a manual is consulted. At this point, manuals may become projectiles as often as they are read and understood. One of the improvements in application development is the incorporation of online help. In many instances, this form of information can lead a user through a difficult task, especially if it is context sensitive. The development of hypertext help systems enable users to browse from one topic to another seeking additional information.

Personalized Environment

Computers have been called cold, hostile machines which have no reflection of their user's personality or preferences. The graphical user environment, esp. on a color workstation, provides a powerful foundation for user personalization. In MPower, the environment can be stylized to meet the preferences of the user including colors, backdrops, mouse characteristics, etc.

Multimedia Components

MPower is the integration of multimedia components, collaboration tools, and HP VUE 3.0 into a cohesive user environment. The components that were integrated into MPower are: HP VUE 3.0 user environment, FAX software to control a connected FAX modem, scanner software to control a ScanJet IIc color scanner, image viewer, audio editor with a stereo headset (with a built-in microphone), RasterOps VideoLive card and software for live video in a window, window capture software, whiteboard for creating simple drawings or sketching on existing images, HP SharedX for real-time collaboration on a network, HP SharedPrint for processing print jobs including automatic switching between PCL and PostScript and spooling print jobs to the printer, multimedia mailer for the distribution of electronic mail with multimedia components including audio and images.

These disparate components have been integrated into a single environment. Access to most of the components is via a slide-up panel from the HP VUE front panel. Access to SharedPrint and the mailer utilized existing front panel controls. A FAX drop zone has been integrated into the front panel in the same way as the printer control.

Client/Server Delivery

MPower with its enhanced audio and EISA bus utilization is best suited to the newer S700 workstations, the Model 715, Model 725, and Model 735, but much of the MPower functionality is accessible from other platforms. An EISA slot is required for the RasterOps VideoLive Card. A fully functional MPower installation will support all of the components of MPower, i.e. FAX modem, ScanJet IIc, CD-quality audio, VideoLive card, and a printer. The typical installation may not have some of these components, e.g. the VideoLive card or the ScanJet IIc. These two items must be installed on the client system and cannot be shared resources in the same way as the printer and FAX modem are. The client and the server can be on the same machine, but the server has additional memory and disk space requirements. The system requirements depend on the applications running on the client and server systems, but MPower requirements can be itemized separately.

There are two options for the client system: client-based VUE and server-based VUE. The difference between these two options is the location for VUE execution. This option is specified during the installation process. MPower has been designed using a client/server model to provide access to multimedia services on lower cost platforms. In order to deliver a fully functional MPower system without requiring additional RAM and disk space for sharable code, the execution of the MPower system is divided between a client and a server system. The client system executes all of the code that requires client-specific hardware or is best suited to executing on the client system, e.g. the VideoLive software or the SharedX client. The execution of VUE on the server, the server-based VUE, provides the advantages of an MPower environment on lower cost platforms. In some cases, esp. when existing hardware exists, this distributed client solution does not utilize the existing hardware resources, e.g. the RAM in the workstation. For these instances, an installation option enables a system to be configured with a client-based VUE. This second option requires an additional 16Mb of RAM and 300Mb of additional disk space on the client machine.

The MPower server system needs to allow for the client systems that may or may not be installed with it. For estimating purposes 6Mb of RAM on the server needs to be allocated for each client system. A typical server, e.g. a 735 with 64MB of RAM can support between 6 to 10 client systems.

One of the realities of distributed network solutions is the growth of memory usage by applications, the operating system, and the user environment. With MPower built on existing technologies, e.g. HP VUE 3.0, the Motif environment, and the X Window System, the space requirements do not go down when MPower functionality is added, but MPower does provide an opportunity for exploring the memory utilization issue and to configure the whole environment for the best usage of RAM and disk space. An MPower configuration needs to incorporate more than just the cost of the components, but also the network and application environment. Since each installation of MPower will involve more that the installation of the MPower code, an analysis of the network configuration and the application environment must also be performed. In the case of the client-based solution, only 100Mb of disk space would be available for applications. In the case of the server-based solution, over 300Mb of disk space would be available on each node for applications.

Table 1: MPower System Configurations

Configuration	Activity	Minimum RAM Configuration	Minimum Disk Usage	Client Swap
Xstation	0%	6 Mb	0 Mb	0Mb
Server-based	25%	16 Mb	250 Mb	40Mb
Client-based	75%	32 Mb	425 Mb	50Mb
Client-on-Server	100%	64 Mb	800 Mb	min. 100Mb

Client Activity is a relative comparison between the four configurations. The metric can be used to understand where the execution of the MPower software is taking place.

The swap space for the client-on-server configuration is 100Mb swap + 50Mb per Xstation user + 20 Mb per server-based user + 10Mb for each client-based user.

An MPower system configuration can be set up in four configurations from X terminals to running both the client and server code on the same machine. The dynamics of these configurations have a direct effect on required resources and the resulting performance. Table 1 is provided to characterize these four configurations.

MPower has been configured as two products to reflect the component costs and the concept of distributed client/server computing. By dividing the software into two pieces, both the computation resources and component costs can be shared. The MPower server contains some components which can be shared by multiple client systems, e.g. FAX server and the PostScript converter. Since these components can be shared, including them in the MPower server focuses the costs on the server rather than charging each user with the costs in the MPower client product. Table 2 will show for each MPower component the delivery vehicle and execution location for each default configuration. When the client is running on the server as in the fourth configuration, the server and client machines are one and the same.

Table 2: MPower Components and Execution Location (* denotes these components are included with HP-UX.)

			Execution	Location	
Component	Delivery Vehicle	XStation	Server-based	Client-based	Client-on-Server
HP VUE Server*	Server	Server	Client	C-on-S	
Online Help Files	Server*	Server	Server	Server	C-on-S
FAX server	Server	Server	Server	Server	C-on-S
FAX composer	Client	Server	Client	Client	C-on-S
FAX browser	Client	Server	Client	Client	C-on-S
Scanner	DeskScan/UX	Server	Client	Client	C-on-S
Image viewer	Client*	Server	Client	Client	C-on-S
Icon Images	Server	Server	Server	Client	C-on-S
PostScript viewer	Server	Server	Server	Server	C-on-S
Fonts & font server	Server*	Server	Server	Client	C-on-S
Audio editor	Client*	N.A.	Client	Client	C-on-S
Audio server	Client*	N.A.	Client	Client	C-on-S
VideoLive client	Client	N.A.	Client	Client	C-on-S
Capture client	Client*	Server	Client	Client	C-on-S
Whiteboard client	Client	Server	Client	Client	C-on-S
SharedPrint clients - sprint & spadmin	Client	Server	Client	Client	C-on-S
SharedX client	Client	Server	Client	Client	C-on-S
X11R5 server with video & SharedX extensions	Client*	Server	Client	Client	C-on-S
SharedPrint server	Server	Server	Server	Server	C-on-S
Mail viewer, composer & server	Server*	Server	Server	Client	C-on-S
MIME converter	Server	Server	Server	Client	C-on-S

MPower has built upon available technologies to deliver a robust, integrated collaborative multimedia environment. Electronic mail can be divided into three distinct areas: the Mail window, the MailViewer, and the MailEditor.
Multimedia messages can be created in two ways:
Outside of the mailer, multimedia files are selected either singly or in a group and then transmitted by dragging and dropping the desired items on the Mail control.

Within the MailEditor, multimedia files can be dragged and dropped into the editor window and then sent in the same fashion.

When multimedia components are dropped into the edit window, they are depicted in the body of the mail message as an icon. This icon object cannot be edited, but it can be copied, moved, or deleted in whole. One can think of these objects as special characters that can be copied, moved, and deleted, but the dot pattern of the character cannot be altered. This allows the creation of multimedia-enriched mail messages that are syntactically correct. In other words, audio, image, and video frames can be interspersed throughout the body of the mail as opposed to attaching these objects at the bottom of the message only. In this way, the reader of the message knows exactly to what these media objects refer in the context of the message. While in the MailEditor the multimedia objects in messages can be viewed or heard by double-clicking the icon. The relevant client will be invoked for viewing or playing the selected object. By making enhancements to standard UNIX email, MPower provides the capability to send mail to other MPower users as well as any user who uses standard UNIX email. Though a user who is not using MPower or a mailer with MIME (Multipurpose Internet Mail Extension) support will not be able to read the multimedia portions of email containing audio, images, or video frames, the message can still be received by the user. In this way, communications can still occur, but text-only messages would be the best media type. As members of the Interactive Multimedia Association (IMA), HP and other members are working towards defining a set of multimedia standards. When these standards become available, HP will be able to offer multi-vendor, multimedia-based mail. In addition to MPower from HP, there are several MIME implementations on other platforms, either pending or available.

The MailViewer window is used for reading messages. Any multimedia object in the message that can be recognized will be shown in iconic form with the appropriate media icon. As in the MailEditor, these objects can be viewed or heard by doubleclicking on the icon. The appropriate client will be invoked for each object. To store the media objects, the appropriate client must be invoked by double-clicking on the icon and then storing the selected object from the client and not the MailViewer. On non-MIME systems, multimedia objects will be receivable in most cases. The text "This is a message in MIME format." will be displayed before the ASCII representation of the multimedia object. Saving this object with a .mim suffix and then opening the resultant file in the File Manager on an MPower system will invoke a MIME viewer and expose the contents of the message in a viewable form.

The transmission of multimedia objects may exceed site specific email file size limits. Since these limits may have been instituted prior to the need for larger file sizes, sending and receiving multimedia messages may require a basic change in a site's email policies. Recognizing these constraints and preparing for the possibility of large messages is critical to the usability and acceptability of multimedia messaging. Even with significant compression, both audio and images have the potential for creating huge files. Audio files require around 16,000 bytes/sec. and a single JPEG video frame takes approximately 90,000 bytes. Scanned images can require several megabytes depending on the resolution, size, and colors specified. A picture may be worth a thousand words, but one must allow for the space required to use multimedia.

Table 3: MPower Standards Supports

Area	Standards Supported
Audio	Mulaw, Alaw, linear8, linear8offset, linear16, RIFF (PC), Sun & NeXT
Image	TIFF (LZW, Packbits, CCITT Group 3 & 4, JPEG), FAX Group 3 & 4, GIF, JFIF, PostScript, & limited XWD support
Video	NTSC, RGB, PAL, SECAM
FAX	CCITT V.33 Group3 with the ability to convert and FAX ASCII, FAX, and all the supported image filetypes.
Printing	ASCII, PCL & all the supported image filetypes
Electronic Mail	MIME/Metamail, SMTP via sendmail
User Environment	MIT's X Window System Version 11 Release 5 (X11R5) & Motif 1.1

Standards Used in the MPower Environment

One of the major strengths of the MPower environment is the utilization and support for industry standards. By supporting standards, barriers to productivity can be eliminated and communication is enhanced. Table 3 documents the current standards that are supported. As standards evolve, MPower will evolve with them and add additional standards as necessary. Standards enable the transmission of information and execution of programs across disparate platforms and environments. MPower will continue to evolve in the area of supported standards. Having information that is not viewable is unacceptable if a standard exists. Each of the MPower components has supported standards to ensure that information can be accessed and transmitted in an accessible form.

Conclusion

The MPower environment is a robust, integrated environment which has been designed to take advantage of the power and distributed nature of workstations. The approach in the design and delivery of MPower has been to integrate multimedia capabilities to increase user performance and effectiveness with increased communications bandwidth. Just as color has enhanced the conveyance of information in text, the inclusion of audio and video in the user environment adds to the potential for communication. The added capability of doing real-time collaboration across a network provides an immediacy of information that cannot be delivered otherwise.

VisiCalc and 1-2-3 are U.S. registered trademarks of Lotus Development Corporation.

UNIX is a registered trademark of UNIX System Laboratories, Inc. in the U.S.A. and other countries.

Motif is a trademark of the Open Software Foundation, Inc.

PostScript is a registered trademark of Adobe Systems, Inc.

ABSTRACT

INTRODUCTION TO NETWORK MANAGEMENT

Speaker:

Roger L. McKee
Graebeards Consulting
581 Manet Terrace
Sunnyvale, CA 94087
(408) 732-2426

Abstract

The size and complexity of computer and telecommunications networks has grown to the point where the need for automated network management systems has become necessary for the efficient operation of those networks. Nearly all organizations are experiencing the increased importance of data communications and are thus forced to face the daunting task of managing a growing infrastructure.

The rapid growth of LAN technology in the last five years has created the need for better management services applied to those LANs. TCP/IP has evolved as the defacto standard for inter networking in open systems. The solution applied to the management of the network is yet another defacto standard -SNMP. This paper will discuss the current management solutions and the Open Systems Interconnect (OSI) Reference Model functions of:

- o Fault Management
- o Configuration Management
- o Accounting Management
- o Performance management
- o Security management

Some of the questions that will be answered include:

What is Network Management?
How does it differ from Networked Systems Management?
What are the objectives of Network Management?
What are the Real-World goals of Network Management?
What is the hierarchy of Network Management needs?
Who are the major vendors of Integrated Network Management Platforms?
What are the vendors providing with the Integrated Network Management Platforms?
What are SNMP, SNMPv2, CMIS/CMIP, and DME?
What should a "good" Network Management Platform do?

PAPER#: 4006

TITLE: "APPROACHES TO THE OPEN SYSTEMS TRANSITION"

TODD HUTTO
DUN & BRADSTREET SOFTWARE
3445 PEACHTREE ROAD, N.E.
ATLANTA, GEORGIA 30326
(404) 239-2000

Changes in business practices and the incompatibility of proprietary systems have forced the need for industry standards and have increased corporate interest in moving to open systems environments. The objective is to make computing systems interoperable and compatible within a heterogeneous computing environment, while providing the basis of an integrated computer/communications infrastructure that addresses both internal and external information needs.

The benefits of open systems are clear. Users have transparent access to information, regardless of its location within the overall computing environment. Systems are vendor independent, which allows the user to choose the best solution for the business requirements from any vendor. And systems conform to standards, which protects investment and provides portability and interconnectivity.

But why have only a comparative few companies made the move from strategic commitment to actual investment in open systems, despite its obvious benefits? For several reasons. First, companies are concerned about protecting existing investments. Second, the industry assumes there are no commercially viable, robust, high end solutions available. This assumption leaves only one foreseeable option--to go with risky start-ups. Third, companies also assume that the UNIX technical environment is not as sophisticated as more established environments. Finally, lack of skills and resources prohibit the change.

Blocked by these concerns and assumptions, companies may not be aware of the choices available that will help them transition to an open systems environment. This presentation will describe transition approaches to help companies move beyond the excuses they use to delay a transition and into action to more quickly gain the benefits open systems provide.

There are three distinct ways to transition systems: re-architecting, re-engineering, and re-hosting. Each requires an analysis of the benefits and disadvantages, the types of corporations best suited to each approach and the software, hardware and human investments required.

Re-architecting

The re-architecting approach allows companies to maintain existing investments while adding functionality through client/server applications. This approach is ideally suited to companies that have major gaps in their existing systems and that want to obtain additional functionality. Companies requiring re-architecting may also have a significant number of databases to migrate.

This approach bridges the old and the new, providing a means to use existing investments with newer functionality. It also allows companies to gain experience in open systems to understand basic open systems principles. And it helps to avoid the incremental costs associated with maintaining proprietary systems.

Additional benefits of this approach include minimal disruption to users and retraining requirements. Companies can incrementally add skill sets, allowing them to reduce disruption and better manage the costs of the transition.

For example, an Aberdeen Group report published a few months ago in the U.S. cited a national stockbrokerage firm that implemented a problem resolution tracking system for service centers, supporting both stockbrokers and sales assistants. The application significantly reduced the time it takes to research and resolve customer and internal problems and was replicated to other sites. The firm also implemented a human resource system that uses a compensation planning database to give managers a consistent planning model for compensation, including year-end bonuses and budgeting.

There are, however, several disadvantages associated with re-architecting. Human, hardware and software costs tend to be higher. Costs may not immediately decrease since systems and skills must be duplicated for a period of time and because multiple licenses are maintained. Also, companies must continue training with current operating systems releases. This approach also adds a level of complexity that may be difficult to successfully adjust.

Re-engineering

Re-engineering is suited to companies that want to make a fundamental change to improve operations and cannot do so by simply putting quick repairs on old systems. Instead, taking risks is the means to significant gain. In general, systems within these companies are not meeting current needs and their disposal is either financially justified or simply mandated. In other cases, hardware or software systems have been orphaned and are no longer supported by the company or the original supplier. Companies well suited to this approach, therefore, require immediate systems change.

The advantages of this approach are, primarily, associated with cost. While initial investment is high, moving immediately to open systems is less costly overall and permits a unified focus on forward thinking development. This allows companies to place new development activity on a single platform rather than trying to maintain two platforms. Often, this approach gives access to innovative solutions since these are frequently incompatible with legacy solutions.

The disadvantages of re-engineering are obvious. First, re-engineering involves abrupt change. It is an "all or nothing" method to gaining open systems benefits, making the strategy of testing pilot programs impossible. Second, it is highly disruptive to user organizations and requires significant retraining. The complexity level in dealing with a new technology cannot be underestimated. Third, this type of move involves risk, because companies may need to turn to lesser-known vendors, including start-ups, that may have solutions immediately. This is a potentially dangerous path. Companies must calculate the potential payback of vendors that may not have the "staying power" required to support business in the future.

Re-hosting

The re-hosting approach allows companies to maintain the functionality they already have but move to a different, open platform. Ideal candidates for this approach include companies that are satisfied with the functionality of their current systems but want to reduce information technology costs. These companies are not willing to risk significant disruption in their organization to gain cost benefits. They are not risk averse but they do require security. In many cases, companies assume they must expect less functionality to reduce IT costs. This is, however, an inaccurate assumption.

The primary advantages of re-hosting are that companies protect investments and endure minimal disruption while reducing the overall costs of the computing platform. They can reduce the load on existing platforms and place functions in more appropriate environments. As a result, productivity improves, existing platforms can be descoped to save costs and new users can be introduced without incurring further costs. A complete move to open systems is achieved through pilot projects and then later through a gradual move, providing better control of change and the learning process.

For example, a Southeastern hospital had disparate hardware and software systems and needed to integrate systems and reduce costs. It found that a mainframe solution was an excessive and unnecessary expense, so it downsized operations. The resulting savings in maintenance and depreciation/amortization expenses saved the company approximately $800,000 per year.

The disadvantages with this approach are similar to those associated with the re-architecting method. Primarily, there is a clear cost associated with straddling two systems--one old and one new. Companies must maintain both and their associated skill sets. They must also continue training with current operating systems releases. Costs may not, therefore, reduce immediately.

Conclusions

It is not easy for companies to make the move to open systems computing if they do not fully understand the advantages and disadvantages of several transition approaches. Awareness of options and an ability to work with vendors that have similar choices will help companies quickly realize open systems benefits and will be prerequisites for success.

PAPER NUMBER: 4008

TITLE: Emerging Trends in Client-Server Development

PRESENTER:

Paul Cubbage
Dataquest
1290 Ridder Park Drive
San Jose, CA 95131-2398
408-437-0292

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Paper No:

4009

Title:

Porting Applications between UNIX and DOS Using the X Window System.

Author:

Don M. Dailey

Address:

Quarterdeck Office Systems
150 Pico Blvd
Santa Monica, CA
90245

Description:

This session covers the whys and hows of porting X Window and Motif applications from UNIX to DOS. With the help of a pre-emptive multitasking, multiwindowing DOS environment that fully supports the X Window System and cross-platform networking, you can run your X clients locally and remotely and even run DOS and Microsoft Windows remotely from X workstations. By integrating your DOS machines into your UNIX network, you can enjoy the benefits of cross-platform computing, giving your users access to a wider range of applications on their existing machines. Because users no longer need both an X workstation and a DOS machine to run X applications, cross-platform computing can significantly reduce hardware costs.

The session will describe the programming environment, libraries, X toolkits, porting issues and porting tools.
A specific porting example will be discussed to demonstrate porting issues and tips.

This session will be of interest to all attendees who run UNIX, DOS X windows or Motif applications.

Session Materials:

Presentation materials will be available at the session.

Paper # 4011

68K Workstation Operating System Strategy

Doug Blackwood
Hewlett-Packard Company
Software Technology Division
Fort Collins, Colorado

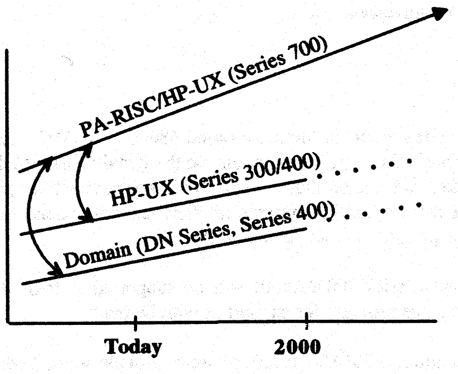
Today, Tomorrow, 2000 ... with 68K Domain and HP-UX

HP's lifecycle for software includes providing extended support and assuring a long term viable operating environment for the systems. Toward this end, the Software Technology Division (SWT), of the Worldwide Customer Support Operations, is responsible for established products. SWT's charter is to ensure that installed base customers continue to receive quality support for software that is in the latter part of its lifecycle. The division currently supports MPE V, RTE, Domain and HP-UX (Series 300/400) operating systems.

Extended Support for HP Software

HP's Software Technology Division, is focused on providing investment protection to customers that maintain software support contracts by:

- Transitioning software to extended support stage at the proper point in its lifecycle
- Providing support - Fixes, enhancement, and interoperability for extended period
- Supporting migration to next generation products by helping to provide a "bridge" to the future (see illustration)



Increased Support for Users of Domain & HP-UX

HP recognizes the diversity of the Motorola-based workstation users (technical & industrial), and is committed to provide ongoing support and enhancements of the Domain and HP-UX OS products.

For HP-UX and Domain this means...

- Current product engineering (maintenance of existing systems, fixes to serious /critical defects, patch delivery)
- Interoperability enhancements (with Series 700, PA-RISC/HP-UX)
- Customer driven enhancements (such as extended peripheral support)
- Improved communication of the availability of software updates, as well as improved distribution of those updates
- Improved methods to receive and respond to product requests or suggestions
- Specials, where the additional investments for unique requirements pay off in extended use of current systems
- Migration paths and upgrade programs customers can use to move to HP open systems over time and at reasonable cost
- Incentives to renew or add support contracts

Domain:

In early 1992, SWT assumed responsibility to provide extended support of the Domain/OS (DN Series & Series 400) through the year 2000. This change has been positively received by the Domain installed base as it provides support from an organization that is able to focus on their particular needs.

This past year SWT worked closely with users via customer visits, surveys, letters, phone calls, and through involvement with users groups (InterWorks, HPWorks, and InterWorks - Japan) to identify and implement necessary enhancements. An upgrade with the enhancements is now available.

Some of the key enhancements are:

1. X11R5 run-time environment
2. Motif 1.2 run-time
3. hpterm 1.3
4. User Environment Development Kit
5. Domain patch process improvements

HP-UX for Series 300/400:

HP has moved responsibility for support of Motorola-based 68000 HP-UX (Series 300 & 400) to SWT. As in the case of Domain, this transition will enable the installed base to drive their needs with an organization dedicated to focus on those needs. SWT has successfully provided focused, customer-driven support for other operating systems to allow users to extend the value of their existing investments. This experience is transferable to HP-UX.

The HP-UX operating system (Series 300 & 400) will be supported at least through the year 2000. SWT will provide the same program for support as with Domain.

SWT will work with the Domain and HP-UX installed base to provide support which will allow users to maintain, mix (interoperate), or migrate their OS investment. The needs of Series 300/400 installed base customers which include a more stable environment, will be addressed. Some specific functionality for Series 300/400 HP-UX that is viewed as critical to system interoperability and migration will be provided.

4012

SoftBench Framework and ToolTalk: A Technical Comparison

Judy Walker and Jerry Duggan

Hewlett Packard Corporation

3404 E. Harmony Rd., Fort Collins, CO 80525

(303)226-3800

Abstract

The Softbench Framework and ToolTalk both offer tool communication services. Both product's messaging services are similar, but the SoftBench Framework also provides features that enable developers to integrate existing applications without source code modifications, as well as offers essential utilities for properly integrating a wide variety of applications.

Introduction

What is a framework, anyway? The answer is, it depends upon what framework you are talking about. On Unix workstations in general, frameworks provide a means for software tools to cooperate under the control of messaging provided by a message server. The common frameworks available on Unix workstations are the SoftBench Framework[1], provided by Hewlett Packard and ToolTalk[2] provided by Sun[3] Microsystems. These two frameworks have similar messaging components but have interesting differences due to choices in vendor focus. ToolTalk's focus remained in the messaging technology areas, while the SoftBench Framework created additional facilities for building distributed applications. Similarities and differences between SoftBench and ToolTalk are the subject of this paper as well as a look into the future to see what a joint COSE (Common Open Software Environment) framework of the future could hold.

First and foremost, ToolTalk as a framework and the SoftBench Framework are not equal. ToolTalk currently provides a C Application Programmer's Interface (API) that is used to add messaging capabilities only to an existing application or when writing new programs. The

---

[1] SoftBench is a trademark of Hewlett Packard Corporation..

[2] Copyright of Sun Microsystems, Inc.

[3] Sun is a registerd trademark of Sun Microsystems, Inc.

SoftBench Framework also provides an API for messaging, but in addition, provides components for controlling processes on another machine (SPC), accessing data on distributed systems (RDA)[4], event loop support and GUI components. Why then, is there such a difference in what two venors call ***framework*** products? SoftBench Tool Developers recognized that having additional facilities for building distributed heterogenous applications was a needed emphasis. As these facilities grew and developed, they became part of the SoftBench Framework . The ToolTalk focus has been to provide powerful messaging with procedural and object oriented interfaces. A true comparison of the two frameworks can legitimately be made only for the component that is the same, messaging. Figure 1 shows the facilities provided by each framework.

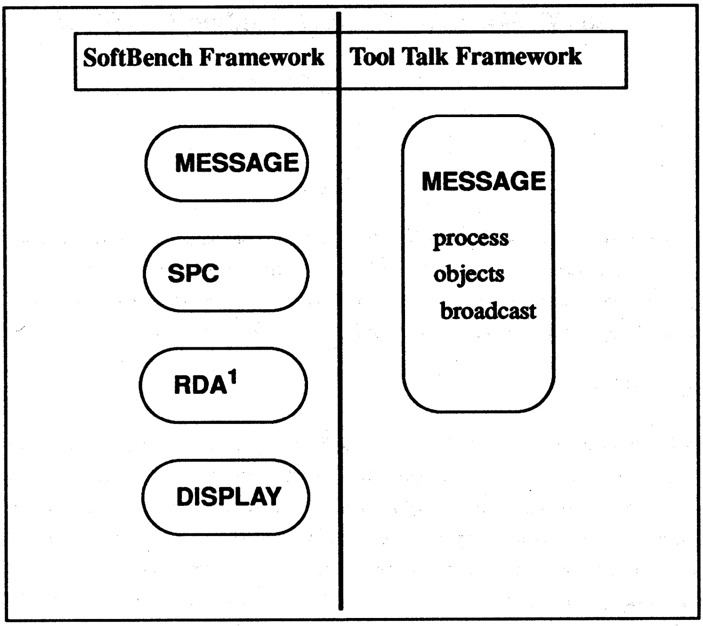


Figure 1. Components of SoftBench and ToolTalk frameworks

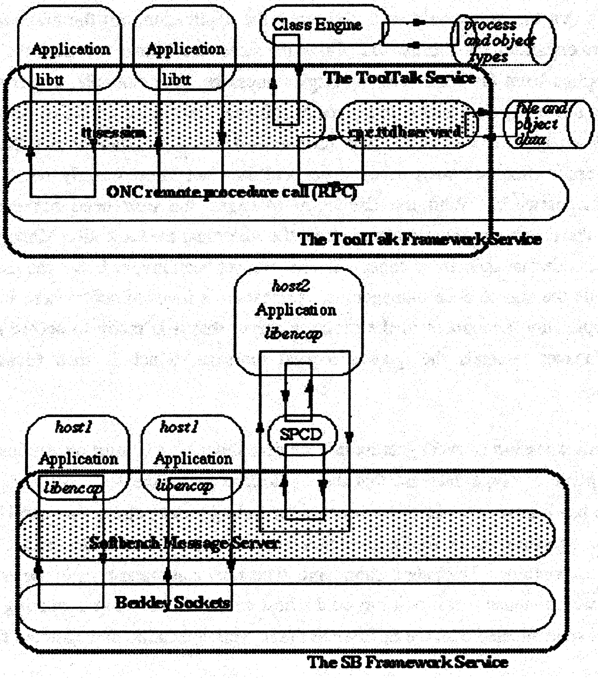
First we will explore the differences in the messaging theme between the systems. Second, we will elaborate on the additional facilities provided by SoftBench. Finally, hints about what the common framework from the COSE effort might provide. A consumer reports style grading will be given to all areas reviewed.

---

[4] ToolTalk has some Remote Data Access via messages only.

Messaging architecture

Both ToolTalk and the SoftBench Framework provide a library enabling client programs to send and receive messages to a separate process called a *session* or *message server.* An API is provided for the client's use and the server maintains the responsibility of tool starting and the routing of messages. ToolTalk is based on ONC's RPC while the SoftBench Framework is based on TCP/IP and Unix domain sockets. Both frameworks have the same messaging model for tool communication. The view is that some messages will be requests for the services of another tool and a reply is expected in response. Request are sent from a tool with little or no knowledge of what tool will be servicing it. Applications that want to service the message register their interest with the server for doing so. This is the basis for a cooperative environment. In addition, some messages are notifications that "interesting" things have happened (FILE-MODIFIED in the SoftBench Framework, for example). Eavesdropping on requests and replies is available in both systems. See Figure 2 for the architecture of each framework.


Application
libtt
Application
libtt
Class Engine
process and object types
The ToolTalk Service
ttsession
rpc.ttdbserverd
file and object data
ONC remote procedure call (RPC)
The ToolTalk Framework Service
host2
Application
libencap
SPCD
host1
Application
libencap
host1
Application
libencap
Softbench Message Server
Berkley Sockets
The SB Framework Service

Architectural Differences

As seen in the previous figure and general description, the messaging components of each framework are very similar. There are subtle differences, however, in these areas: 1) The way that integrated tools are started from the server. This is known as execution management. 2) How integrated tools are described both statically and dynamically. 3) How integrated tools are addressed and scoped.

Execution management

The SoftBench Framework maintains the tool environment automatically via the Execution Management facility. The Execution Manager is responsible for invoking tools when a message is sent and there is not an appropriate tool already running to handle that request. This means that the user need not be concerned with whether or not a tool has been started in the correct place in the network to handle a task that is desired. The Execution Manager uses an *execution* string to know how to start the appropriate tool somewhere in the network. The string looks much like the command line that the user would enter to invoke the application, but the *execution* string also contains macro expansion capabilities that allows the Execution Manager to start the encapsulation with data supplied from the corresponding request message. For example, a user may be in the editor about to make an enhancement to a piece of software, but the file must first be checked-out of a source code management system. The user doesn't know whether or not the source code management application has been invoked yet and may not know exactly where it should be invoked in the network. With the Execution Manager, the user need not ever know this information. The check-out request is made from the editor and the Execution Manager determines that the source code management encapsulation has not yet been invoked. So, the request message is queued while the source code management application is invoked somewhere in the network. When that application announces to the message server that it is ready to accept messages, the Execution Manager re-sends the queued request message, which is then forwarded to that encapsulation.

The ToolTalk ttsession (server) gets its information about how to start an application from the application's ptype or otype database that the application writer created, compiled and installed. This database has a large amount of information including a *start* string that can be used by the ttsession if the ttsession cannot find a running version of the application to send the message to. This *start* string contains a Unix shell command. The *start* command runs in the environment of the ttsession, which means that it will run on the host where the ttsession is running only. This is somewhat less sophisticated than the SoftBench Framework execution management facility.

Addressing

One of the primary functions of both message server implementations is to determine which process is going to be selected to service a particular request. Both systems encourage the sender of a message to know as little possible about the receiver, and let the default delivery mechanism determine the receiver. Tools that want to service a message register their interest with the server. ToolTalk provides additional "addressing" modes beyond this. The first is to have the sender explicitly indicate which tool will be the receiver, and is referred to as *process addressing*. The second addressing mode ToolTalk provides is an object based delivery mechanism, which uses the object specified (actually, the type of the object specified) to determine the receiver of the message. The SoftBench Framework allows "broadcast" messaging, that is, some messages (such as STOP) can be sent to all tools currently running that have express interest.

Scoping

The SoftBench Framework allows multiple sets of cooperating tools within a single running instance of the server. These tools are grouped according to the *project* they are currently working on. This project is always associated with a Unix filesystem directory, which provides a convenient storage location for project specific data. For example, a build tool might have different sets of options if its working on one collection of sources vs. another.

ToolTalk has a different concept of "scope". Messages can be session scoped, file scoped, 'both' scoped, or file-in-session scoped (the latter two are the conjunction and disjunction of the former two). These specify session level tool cooperation or tool cooperation based on the interest in a specific file. The latter is supported internally across sessions. In other words, regardless of which ttsession an application is linked to, it will hear messages from any other tool scoped to that file. This is more difficult to accomplish with the SoftBench Framework since its cooperation is centered around projects using the same message server. ToolTalk definitely has an advantage here. Interest in files is common among users on different systems using separate ttsessions. For example, file modification is an event that many users would be interested in regardless of what ttsession their editor is linked to. A SoftBench Framework with a single global project would be equivalent to session scoping in ToolTalk. The SoftBench Framework doesn't directly have an equivalent to file scoping, but this can be implemented using multiple message server connections (see below). Figure 3 shows the project scope model supported by the SoftBench Framework and the ToolTalk scoping model and the session/file view of the ToolTalk model.

Tool Descriptions

ToolTalk allows tools to statically describe the services they provide and the notifications they listen to (via ptypes and otypes). The static description of ToolTalk also offers additional features above its API that can set up constraints based on the type of tool. For example, only so many processes of a certain type can observe a particular file. The SoftBench Framework does not provide a static description to the granularity that ToolTalk does. SoftBench statically lists the Toolclass type and a set of services is implied by that toolclass. SoftBench also provides a way to dynamically query which requests a tool can service.

Multiple Connection Models

The SoftBench Framework provides a general-purpose mechanism to allow a message object to connect to any SoftBench message server. Processes can have multiple message objects open at once to the same server or to different servers. This is useful for implementing *super tools* in the first case and in implementing server *bridges* in the second case. Super tools are tools that have been identified as a specific tool, but control sub tools within the same process. Both Softvi and Softedit have used this concept. Bridges allow an application writer to put together arbitrary networks of message servers. ToolTalk advocates only a single connection to the message server, thus restricting the flexibility of the application writer. This is reviewed in the API discussion.

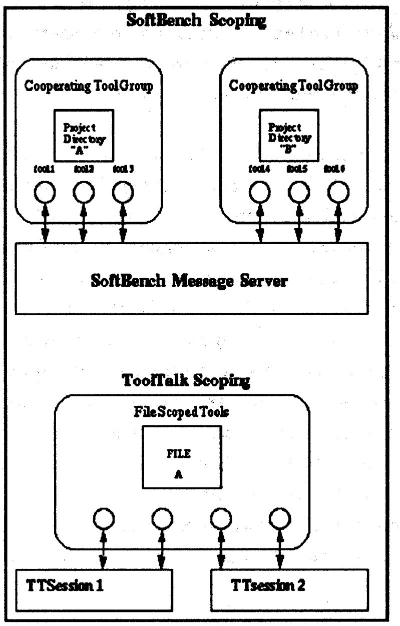

SoftBench Scoping
Cooperating ToolGroup
Project Directory "A"
tool1
tool2
tool3
Cooperating ToolGroup
Project Directory "B"
tool4
tool5
tool6
SoftBench Message Server
ToolTalk Scoping
FileScopedTools
FILE A
TTSession 1
TTsession 2

Figure 3. Scoping conventions of the SoftBench Framework and ToolTalk

Object Oriented Features

The SoftBench Framework allows overloading based on the type of the file specified in the message. In other words, a message with the same toolclass and operation name may be dispatched to different tools based on the type of the operand referred to in the message. ToolTalk has more sophisticated OO features. Each file can have additional "object Specs" associated with it. These object specs can be thought of as object instances, and are basically collections of properties. There can be many object specs associated with a single file. The properties are not stored in the file itself, but rather in an associated database. This causes problems if the original file is removed, copied, or renamed. ToolTalk provides library calls and shell utilities to perform these functions correctly.

Object type definitions and methods are described in an "otype" file. This file describes the methods available on an object via "signatures," which associate an operation and arguments with a particular tool. It is possible for a given object type to inherit methods from other types, but this inheritance is specified on a signature by signature basis.

Object based messaging in ToolTalk is performed as follows. The sender of a message fills in the operation name, the object id (object reference) and any arguments. This message is sent to its ttsession. The server gets the otype of the object id, and uses the operation name and arguments to determine the receiving tools. It then forwards the message to these receiving tools. Since this lookup is based on the type of the object, overloading is supported.

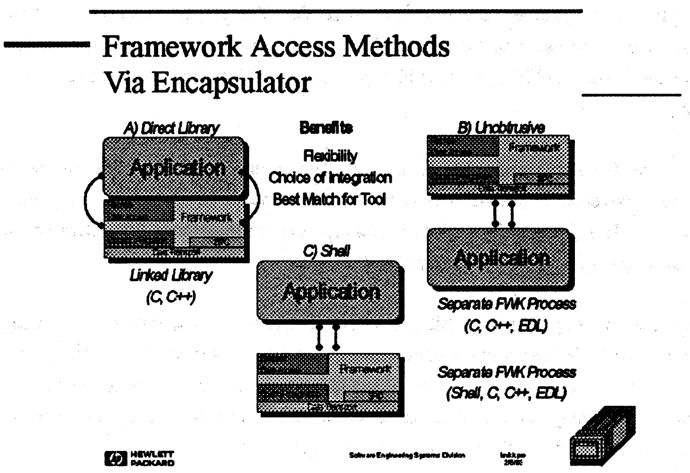
API Differences

In the last section, a description of the differences between architectural features was elaborated upon. There are differences in the API as well. Some of these differences reflect the architecture mentioned in the former section. Other differences are due to the vendor's chosen level of exposure. The SoftBench Framework provides an easy to use interface that is consistent for the many components provided by the framework. There are also several different access methods to the messaging component. ToolTalk's API is slightly lower level with a focus on message structure manipulation. It is important to understand the basic differences between the two framework APIs before moving on to detailed code examples. Code examples for each type of messaging activity are shown for both ToolTalk and the SoftBench Framework.

General Description

SoftBench Framework

The SoftBench Framework provides multiple access methods for writing integration software. These methods are called direct library access, unobtrusive access, and shell access. (see Figure 4) Direct library access is used when calls to an archived or shared library are added to an application. This API is called ***Encapsulator***. Unobtrusive access is used when an application has some form of programmatic interface, either textual, through sockets, remote procedure calls, or through some other text passing mechanism. In this case, no source code modifications to the application are necessary, only access to the binary program is required. Shell access is provided so that shell scripts can be written to access the message server. Shell access is also used by programs such as 'emacs', that have the ability to execute a process. By providing multiple access methods, integrators can choose the method that best suits their application.



Framework Access Methods
Via Encapsulator

A) Direct Library

Linked Library
(C, C++)

Benefits

Flexibility
Choice of Integration
Best Match for Tool

C) Shell

B) Unobtrusive

Separate FWK Process
(C, C++, EDL)

Separate FWK Process
(Shell, C, C++, EDL)

Figure 4. Access methods available from Encapsulator

The Encapsulator provides an API for messaging that reinforces the project model for applications. Message connections are made by creating a message ***object*** that has scoping characteristics specified as ***attributes***. ***Events*** are declared for registering interest in specific messages in the same project. Attributes and events are added to an object and become part of the object. These objects are used for sending messages as well as receiving them. All the information needed to match an incoming request is available in the attributes and events of the Message Object. Messages are never created or destroyed. That part of the implementation is handled for the user which keeps the interface simple. In all, there are less than 50 functions needed to supply messaging to a program.

SoftBench messages themselves are string fields separated by whitespace. Pattern matching of message events received by a tool is therefore string based. The fields of the message string represent the address, toolclass, action and data.

ToolTalk

The ToolTalk approach provides data structures called *patterns* and *messages*. The API supports full manipulation of these data structures. Explicit calls to set up and close the connection to the server, *ttsession* are provided as well. Functions are provided for memory management, but the responsibility of that management falls on the shoulders of the application writer. The API has over 250 calls.

The **TT_pattern** structure is used to provide the server with information about what messages the tool is interested based on the ToolTalk model of scope, address and type. Arguments of various types are allowed to further restrict what patterns are matched. Allowing multiple separate arguments for pattern matching has a definite advantage over the string pattern available in SoftBench. Not only can you match on different types, but string patterns with embedded whitespace may be matched as well.

The other data structure, **TT_messages**, is the structure passed logically from tool to tool. This structure holds all of the information for the state of the message, the pattern matched, user data, and so on. The API for manipulating this structure provides creation, value setting, value retrieval and message destruction. The same message structure is passed around to an interested party, examined, the sent back with a modified state as the reply. This differs from the SoftBench Framework which never gives you a handle to the actual message.

Tooltalk has little support for multiple connections in its API. The file descriptors and connection information are held in global variables that may be manipulated. This is error prone and becomes painful to keep track of.

Joining a Server

Both ToolTalk and the SoftBench Framework provide a means for starting the processing of Message traffic.

ToolTalk

These functions are required to create a connection (open a file descriptor) and specify which ToolTalk message server is joined. A global variable for the session id is set with this function.

```
 my_procid = tt_open();
 ttfd = tt_fd();
tt_session_join(tt_default_session());
```

SoftBench Framework

The message server of interest is specified as an attribute of the message_object when the object is created. The channel is opened during creation as well.

```
message_object = encap_make_object(NULL, "MO2",  encap_Message,   0,
         MSERVE("somehost:id"),  event1,
         event2,event3, 0);
/* This function is required to start the message traffic. */
 encap_start_message_connection(message_object);
```

Opening Multiple Channels to the Server

ToolTalk does not advocate multiple connections to a ttsession, or multiple connections to different ttsessions in the same process. Trying to accomplish multiple connections to the ttsesssion with the ToolTalk API exposes the fact that it uses many global variables, such as the ttsession and the channel id to the ttsession. It is possible to get multiple connections, but it means that the user will have to manage the state of these variables (via functions that manipulate them) when attempting this. Message Objects naturally support multiple connections to the same or different SoftBench message servers.

ToolTalk

Multiple connections to different message servers can be achieved in ToolTalk by calling tt_open() and tt_fd() multiple times. The session specification is done via the tt_X_session call with a host specification. These file descriptors returned may be added to the application's event loop. (For example XViews call notify_set_input_func().) Multiple connections to the same message server may be accomplished if the global procid is maintained by the user.

```
* Initialize ToolTalk, using the initial default session, and
* obtain the file descriptor that will become active whenever
* ToolTalk has a message for this process.
*/

    my_procid = tt_open();
    ttfd = tt_fd();
    tt_session_join(tt_default_session());

    /* Add additional connection to a session on hpfccg */
    my_session = tt_X_session("host2:0");
    /* This resets the global var holding the ttsessionid */
    tt_default_session_set(my_session);
    my_procid = tt_open();
    ttfd = tt_fd();
    tt_session_join(tt_default_session());
```

Softbench Framework

Message connections may be specified to be made to different SoftBench Frameworks when the Message Object is created via the MSERVE attribute for the Message Object. This attribute carries the id for the message server of interest and defaults to the local message server.

```
message1 = encap_make_object(NULL, "MO1", encap_Message, 0, 0, 0, 0);
message2 = encap_make_object(NULL, "MO2", encap_Message, 0, MSERVE("anyhost:id"), 0, 0);
```

Handling of Events

ToolTalk returns access to the file descriptor of the message channel. The application writer would add this file descriptor to the native window system event loop's list of inputs, or *select's* event loops list of inputs. SoftBench Encapsulator provides an event loop for automatic dispatching of events for all of encapsulator object types. These include message objects, display objects, subprocess control objects, and signal objects . This event loop will call either the Encapsulator event loop or the event loop of the native window system depending upon how the event loop interface is set up.

ToolTalk

This is an example of a function that enables ToolTalk input to be added to an application. It is provided by XViews in this case.This event loop function is provided by XViews.

```
notify_set_input_func(base_frame, (Notify_func)receive_tt_message, ttfd);
xv_main_loop(base_frame);
```

Softbench Framework

Encapsulations require the encapsulator event loop function, encap_event_loop, to be used. This means that under certain circumstances, wrappers are needed around the functions of the native window system that add input file descriptors, exception handlers, remove input handlers, etc.

```
ncap_add_input_handler = add_input_wrapper;
encap_dispatch_event = main_loop_wrapper;

event_loop();
```

ReceivingMessages

In tooltalk, pattern structures are created that specify what type of message the application is looking for. Specifying a message of interest is done by adding to the pattern structure restrictions for what kind of message you are looking for. Scoping can be changed to be more like the SoftBench Framework's *project* by adding arguments to the pattern that specify toolclass, and directory context.

Callbacks can be specified for direct calling when a matching message occurs using the tt_pattern_callback_add function.

ToolTalk

```
pat = tt_pattern_create();
tt_pattern_category_set(pat, TT_OBSERVE);
tt_pattern_scope_add(pat, TT_SESSION);
tt_pattern_op_add(pat, "ttsample1_value");
tt_pattern_register(pat);
tt_pattern_callback_add(pat, somefunction);
```

SoftBench Framework

When an incoming message of interest occurs, an *instance* structure is passed in providing the infomation needed about the message. Using a ReplyTrigger in Encapsulator is analogous to setting the TT_OBSERVE category in ToolTalk. Encapsulator provides a means to have events directly routed to the callback function when a match with the context, action and type (reply or notify) of message occur.

```
r_trigger = encap_make_reply_trigger( 0,    /*toolclass*/
                                             0,    /*host */
                                             0,    /*dir */
                                             0,    /*file*/
                                             "ttsample1_value", /*data*/
                                             0);   /*status*/

ev = encap_make_event(encap_ReplyTrigger, r_trigger,  receive_tt_message, NULL);

encap_add_event (message1, ev);
encap_free_reply_trigger (r_trigger);


receive_message(encap_object obj, encap_instance inst, char * data)
{
  xv_set(gauge, PANEL_VALUE,  atoi(encap_get_instance_data(inst)), NULL);
  return;
}
```

Sending request messages

In the ToolTalk world, messages are the structures that should be created and filled appropriately with the right address, state, data, and specific matching information. Callback functions can be added to the message. These callbacks are invoked when the message is sent and again when the message returns with a reply. An application sending request messages to itself requires careful understanding about the callback execution sequence. Callbacks for messages are exectued before callbacks for patterns. In other words, the callback for the send command will be executed before the callback for the service. The state of the message will be available in the message in order to tell when to actually provide service.

SoftBench's send_request function sends the request and catches the response to that request. Even if the request is caught by the same process, the callback associated with the response is not invoked until the request has been handled, and a response has been given.

ToolTalk

```
Tt_callback action
cntl_msg_callback(message, pattern)
{
 mark = tt_mark();
 switch (tt_message_state(message))
          case TT_STARTED:  break;
          case TT_HANDLED: /* return the replydata */
```

```
                                    tt_message_arg_val(message, index_number);
                                    break;
            case TT_FAILED:     break;
            default:
    tt_message_destroy(message);
    tt_release(mark);
    return TT_CALLBACK_PROCESSED;
}

Tt_status
send_a_message(char * action)
{
  msg = tt_message_create();
  tt_message_op_set(msg, action);
  tt_message_address_set(msg, TT_PROCEDURE);

  tt_message_arg_add(msg, TT_IN, TOOLCLASS, toolclass);
  tt_message_file_set(msg, file);
  tt_message_callback_add(msg, cntl_msg_callback);

  result = tt_message_send(msg);
  return result;
}
```

SoftBench Framework

Encapsulator sends the message out on the channel of the specific message object. All tools listening within the scope of that message object will have access to the message if desired.

```
void
cntl_msg_callback(encap_object obj, encap_instance inst, void * data)
{
  encap_string   data_string;

  data_string = encap_get_instance_data(inst);
  /* process the data */
}

send_a_message(encap_object obj, char * action,  char * data)
{
  /* pattern matchin inforamtion has already been placed in message object. */
  encap_send_request(obj, TOOLCLASS, action, Null, data, cntl_msg_callback, user_data);
}
```

Sending notify messages

ToolTalk

Tool Talk requires creating a message and setting up the attributes within the message. Message space allocation is destroyed after the message is sent.

```
void
broadcast_value(item, event)
Panel_item      item;
Event           *event;
{
  Tt_message msg_out;
  Tt_message msg_out;
  msg_out = tt_pnotice_create(TT_SESSION, "ttsample1_value")
  tt_message_arg_add(msg_out, TT_IN, "integer", NULL)
  tt_message_arg_ival_set(msg_out, 0, (int)xv_get(slider, PANEL_VALUE));
  tt_message_send(msg_out);
  tt_message_destroy(msg_out);
  0);
}
```

SoftBench Framework

Encapsulator sends the message out on the channel of the specific message object. All tools listening within the scope of that message object will have access to the message if desired.

```
void
broadcast_value(item, event)
     Panel_item     item;
     Event          *event;
{
  char buff[200];

  sprintf (buff, "%d\n",
        xv_get(slider, PANEL_VALUE));

  encap_send_notify(message1, /* msg object*/
             "ttsample1_value", /* action */
   NULL, /* file */
   buff, /* data */
   "PASS", /* status */
}
```

Additional SoftBench Framework Components

As mentioned earlier, in addition to providing a programmer interface for messaging, Encapsulator provides convenience routines and other services that are frequently needed when creating or modifying a distributed application. These facilities are presented in a consistent and easy to use manner along with the interface to messaging. These are:

- ***A general subprocess interface that captures the output of the subprocess.***
- ***Operating System interface facilities.***
- ***Remote Data Access utilities***
- ***A general event loop facility.***
- ***User interface library based on Motif widgets, including a cursor-addressable terminal widget.***

Sub-Process Control

The SoftBench Framework provides an abstraction of the Unix process model, called Subprocess Control or SPC. This abstraction allows creation of subprocesses, I/O to subprocesses, signals, and notification of the termination of subprocesses. This abstraction allows a single process to control multiple subprocesses. A major feature of SPC is the ability to create a remote subprocess on a different machine than the one on which the parent is running. The remote machine does not have to be of the same architecture as the "local" machine, and doesn't even have to be running Unix.

The Encapsulator provides two kinds of access to this facility. One is a synchronous use of it and the other is asynchronous use. The API for synchronous use is a simple function, while the asynchrous use of controlling sub-processes uses the Encapsulator's object model. This enables multiple subprocesses to be controlled within the same application. The API provides a choice of type of connection to the subprocess via pipes or ptys. When a process is executed remotely, a sub-process control daemon handles the connection from the SPCD to the applciation in the manner specified by the user. See figure 5.

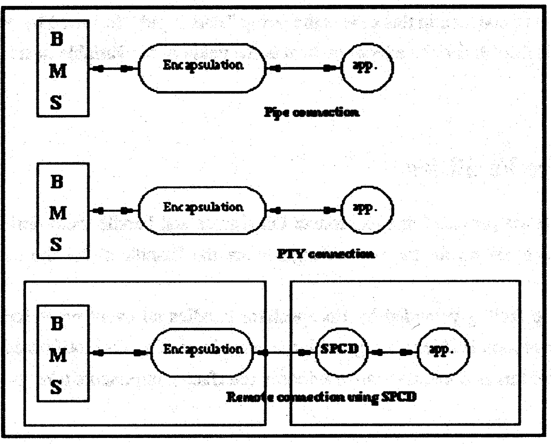


Figure 5. Sub-process Control capabilities

Synchronous distributed process control

```
/* This function always uses a pipe for the connection with the application */
 string system (command, host)

system(string_concat("spell ", path), host);
```

This function will run a shell command by forking a shell first, running the command and then waiting until the command has finished to return its output. The second parameter allows the application writer to specify the host upon which the command is run.

Asynchronous distributed subprocess control

Subprocess objects provide asynchronous distributed subprocess control. Events may be defined for this object. These events are used to match pieces of the stdout or stderr of the process when the subprocess produces it, thus the events are asynchronous. One or more subprocess objects may be introduced into a program.[5] This object is used for unobtrusive encapsulation. If the application being encapsulated reads from standard in and writes to standard out, or is a terminal based application, it can be encapsulated.

```
attribute a = merge_attribute(COMMAND("/bin/sh"),EXECHOST(host1));

evnt = make_event(Application,"This is pid ([0-9]+)",app_callback,
            (void *)"NOTDONE");
spc_obj = make_object(NULL, "SPC_OBJ", Subprocess, NULL, a, evnt, 0);

start_process (spc_obj);
```

The pattern being matched in this case is the string "This is pid " followed by one or more digits. The subprocess is /bin/sh and the execution host is the value of the variable host1.

OS Interface Facilities

Signal objects are provided to synchronize OS signals and handle them during the event loop processing. Events set up for the various signals are the "handlers" for the incoming signal of interest.

The event loop facility provided by Encapsulator handles all event types for all Encapsulator objects. This event loop is able to merge with the event loop of a non-motif based application such as XViews. When this is done, the native window interface components take the place of display objects.

Signal objects are also available for creating synchronous signal handling. The signal object provides an easy means of catching the signals that are sent to the application.

```
object sys_obj;
event sys_ev;

void catch_signal(object obj, instance inst, void *user_data)
{
printf("Signal %s received\n", get_instance_sigval(inst));
}
```

---

[5] This object is created in the same fashion that the Message Object is created. Likewise, events are set up identically to the Message Object's events.

```
sys_obj = make_object(NULL, "SystemObject", Signal, NULL, NULL);
sys_ev = make_event_callback(System, "SIGINT", catch_signal(), "SIGINT");

add_event(sys_ev, sys_obj);
```

Remote Data Access

The SoftBench Framework is designed to support distributed environments, consisting of applications running on different hosts. These applications will have to share data. The SoftBench Framework supports this data sharing via the Unix filesystem and NFS. NFS, however, does not require that the filesystem layout (mount points, etc.) be consistent among different hosts. Thus, a file which is accessed on one host by one pathname might have to be accessed by a different pathname on a different host. The SoftBench Framework supports the correct communication pathnames between different applications, even if they are running on different hosts. It does this by providing routines to translate from a pathname into a *Network Independent Name*, which can be transferred to any host. This Network Independent Name can then be translated back into a pathname which is valid for that host.

In this example, the data that is being sought is in a directory relative to the project of interest specified by the Message Object passed in.

```
path = make_filename(message_object, host, dir, operand, True);
```

The last field specifies whether it is important or not that the file exists.

COSE and the Future

The COSE (Common Open Software Environment) effort will change messaging based tool communication for the better. Some of the proposed solutions will capture the best elements from both SoftBench's and ToolTalk's framework products. The expectations are that distributed features from the SoftBench Framework will be added, and the static tool descriptions and object messaging from ToolTalk will be merged. An API that is easier to use and follows the SoftBench Encapsulator's object model is being proposed in an attempt to make the COSE framework flexible and easy to use.

Table 1: A consumer reports style comparison of the SoftBench Framework and ToolTalk.

Description of service or item	SoftBench	ToolTalk
Remote Procedure Call / Notifications	●	●
Meta Messages: Message Server Control Message Object filtering	 ● ●	 ● ●
Addressing	◍	●
Scoping	●	◍
Tool Descriptions: Static Dynamic	 ● ●	 ● ●
Cross Message Server models: File-scoped routing General	 ● ●	 ● ●
Execution managagement: Distributed features Start strings	 ● ●	 ▨ ○
Object-oriented messaging Polymorphism Inheritance Attribute Management	 ▨ ● ●	 ◍ ◍ ◍
Subprocess Control	●	●

Table 1: A consumer reports style comparison of the SoftBench Framework and ToolTalk.

Description of service or item	SoftBench	ToolTalk
Remote Data Access	●	▨
Ease of Use	●	○
Unobtrusive encapsulation	●	●
OS interface features Signal Handling Event loop control File descriptor access	 ● ● ●	 ● ● ●
Shell Access	◍	●

● ◍ ○ ▨ ●

Best ⟷ Worst

PAPER NUMBER: 4013

TITLE: X For the Non-Expert: A Tutorial on The X Window System

PRESENTER: David Harris
The X Business Group Inc.

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Tuning the HP-UX Series 800 File System

Interex '93 Paper #4014

by

Glen Johnson

Computer Solutions , Inc.
120 East Marks St. #225
Orlando, Florida 32803
407-649-1407 or 512-343-6634

The HP-UX high performance file system (HFS) creates a lot of structure on your disks in an attempt to provide a flexible and fast data storage and retrieval environment. For many applications this structure may decrease instead of increase system performance. This presentation will provide a limited description of the HFS structure, concentrating on the areas we, as users, have some power to control. On top of the HP-UX file system, applications add other files structures. In many cases the files use an indexed sequential access method (ISAM) to store and retrieve information. The last section of the presentation will explain how to manage these files. Only the series 800 HFS will be discussed, but much of the discussion also applies to the other HP-UX platforms.

Basics

Much of the following discussion will concern file sizes. The sizes are measured in bytes. A byte is just a character, such as a letter or digit. A kilobyte is a thousand bytes, but this is a computer thousand. A computer thousand is 1024, not 1000. This is because computers count in base 2 instead of base 10 and 2 to the 10th power, which is the first power of 2 that results in a number greater than decimal 1000, equals 1024. The letter K is used as shorthand for kilobyte. Similarly, a megabyte is a computer million bytes, 1024 times 1024, and a gigabyte is a computer billion bytes, 1024 times 1024 times 1024. The letters M and G are used as shorthand for megabyte and gigabyte.

Blocks

The HP-UX disk management system divides the disk into fixed length number of bytes called blocks. Historically UNIX block sizes have been 512 or 1024 bytes. The Series 800 HFS, however, permits the user to choose a block size of 4K or more. The rational behind the use of a smaller block size is that most files on computers are less than 10K bytes long. If a large block size is used, more disk space would be wasted. For example, if a file is 2000 bytes long, the smaller block size would only require 2048 bytes of disk space (two 1K blocks), while the 8K block size system would require 8192 bytes (one block).

The problem with the smaller block size systems is that even if 80 percent of the files on your system are less than 10K bytes long, the other 20 percent of the files are accessed 80 percent of the time. It is the large files that your programs continually use. As we will see later, the larger the file the longer it takes to access the data.

Files

From your exposure to other computer systems or even if you write COBOL programs, you may be familiar with the concept of relative record files. These are files that are broken into fixed length records that are accessed by the record number. All HP-UX files are essentially relative record files with a record size of 1 byte.

A file, to the file management system, is just a bunch of bytes. It doesn't care that the byte values are or if they are related to each other. It doesn't even care if the bytes are written in order. The program just tells the file system it wants to write X number of bytes to location Y of the file. If a program opens a new file and writes one byte to location 1 million only one block is allocated to the file. The HFS does not add any structure to the internals of the file.

File System

HP-UX does add structure to the disk in order to help manage your files. This structure is called a file system. A file system can be a portion of a disk, an entire disk or portions of multiple disks. The maximum file system size on HP-UX systems is four gigabytes. Most other UNIX systems only permit files systems to be two gigabytes. This is due to the basic design of the UNIX file system. Some of the structures associated with the file system have a field that contains the byte offset into the file system. Since this field is 32 bits in length and is sign sensitive, only 31 bits can be used to represent a byte location. Two to the 31st power is 2 gigabytes. Since a file system can only be 2G, the largest file on a UNIX system is only 2G. Even on the HP-UX system, the largest file is limited to 2G. This means that if your programs use 500 byte records, you can at most have 4 million records in the file. By using a data base subsystem that uses character compression, this limitation can be circumvented. I have encountered a situation where I wanted to convert a file from the TISAM file structure to C-ISAM. The problem was that the file contained 5 million 1000 bytes records. Using TISAM, which has compression, the data file only took up 800 megabytes. The C-ISAM data file would have required 5 gigabytes. We ended up staying with the TISAM structure and using some other method of solving our problem (we wanted to access the data using a 4GL).

Inode

The structure that keeps track of all of the file information you see when you perform a ls -al command is called an inode. The inode contains file information, such as, access and modified times, size, access rights and who owns it. It also has fields for the blocks allocated to the file. Since the inodes are fixed length structures there are a limited number of fields available for pointing to blocks. Specifically, in HFS, there are 14 fields. Since files must be permitted to grow up to 2 gigabytes, some of the 14 fields are used to point to blocks that contain the block numbers allocated to the file.

On other UNIX systems that use 1K blocks, the first 10 of 13 fields contain the actual block numbers of the blocks that contain the first 10k bytes of data. The eleventh inode block field contains the block number of a block that contains the pointers to the actual data blocks. Since a block is 1K long, and the pointers are 4 bytes long, 256 block numbers can be stored in 1K block. Therefore, the eleventh inode block field identifies bytes 10K through 256K of the file. These bytes are accessed by what is known as a single indirect access. Byte 256K plus 1 of a file requires the twelfth inode block field.

This field contains a block number of a block that contains block numbers of blocks that contains block numbers of blocks that contain actual file data. This is a double indirect access and is used for all bytes in the file greater than 256K and less than 64M. Therefore, in order to read or write byte 256K + 1 of a file, three disk I/Os must be performed instead of just one.

Any byte greater than 64M requires four disk I/Os. The thirteenth inode block field is used for these bytes. This field contains a block number that contains pointers to more double indirect blocks. Thus, the thirteenth inode block field is a triple indirect pointer. The reason there are only 10 direct pointers is because when the designers of UNIX looked at their existing systems they determined that the large majority of files were less than 10K bytes long. Therefore, they wanted to make access to those files as fast as possible. Also, if the block size was larger than 1K, too much disk space would be wasted. Remember, these decisions were made over a decade ago when disk space was a very big concern.

This design made access to the majority of the files on file systems fast at the expense of the few large files. But, it is the large files you use more than the small. Your customer master and inventory files, I assume, are much larger than 10K. For this reason, HFS permits you to define the file system block size. Series 800 HFS even forces you to pick a good size (4K or larger) so all of your files can be accessed via at most a double indirect pointer. This is a big advantage of using HFS over other UNIX file systems. To get around the problem of wasting up to 8K-1 bytes of disk space at the end of each file, the last block allocated to a file, that is less than 13 blocks long, uses a user defined fragment of a block.

Fragment

A fragment may be 1K, 2K, 4K or 8K long. If a file is 8K+1 byte long, the first 8K bytes will be placed in a block and the last 1 byte will be placed in a fragment. If the fragment size is 1K (the default), then only 9K bytes of disk space will be used by the file instead of 16K. The other 7K of the block the fragment occupies will be used for fragments of other files. This algorithm permits you to have your cake and eat it too. You have faster data access due to the large block size and you don't have much allocated but unused disk space.

Cylinder Group

The HFS has another feature other UNIX systems don't which is the concept of a cylinder group. On other files systems all of the inodes are kept at the beginning of the file system and all of the free blocks are kept in a global free list. When a file is opened, this structures forces the disk to go to beginning of the file system to read the inode and then go some place into the file system, possibly all the way to the end, to read the first block. The free list keeps track of all of the blocks that are available for use. This starts out as a nice orderly list but as files are added and deleted from the file system, the order of the blocks within the free list can become very disorganized. It is not unreasonable for the first block of a newly created file to be near the beginning of the file system and the next block near the end. The cylinder group structure corrects these problems.

In the HFS, every X number of disk cylinders is handled like a miniature file system. X is a user definable number between 1 and 32 inclusive. Each group has it's own set of inodes and manages its free space. When a directory is created it is placed in the cylinder group that has the greatest number of free blocks and the fewest directories. When files are created, they are placed in the same cylinder group as their parent directory (if possible). This algorithm ensures locality of reference for small files. That is, the internal structures required to manage a file system and the files data are physically close to each other on the disk. This makes stand alone access to the files data fast.

The other problem that HFS corrects concerns the handling of free space. Instead of a global free block list, each cylinder group has its own free block bit map (an array of words in which each bit represents a block in the cylinder group). The use of a bit map ensures that when a file grows and requires another block, the "best" block can be allocated. The best may or may not be the physically next block depending on a user definable parameter and what is available. But, at least some intelligence will used to make the choice instead of just using the next block on the free list.

Directories

Directories are special files that would rarely require the indirect pointers of its inode. On most UNIX systems that use a 1K block, directories contain 16 byte entries. Two bytes contain the inode number of the file and 14 bytes contain the actual file name. Each block can thus hold 64 entries. As files are added to the directory the next available entry is used. Directory management is a sequential process. To add the 577th [(9 * 64) + 1] file to a directory requires the first 9 blocks to be read and searched for an empty entry before a new block is allocated for the directory and placed in the tenth inode block field. If there are 641 files in the directory, the last file would require an indirect block access to get the directory entry. Therefore, just to find the inode of the last file would require 14 disk reads (10 direct blocks, 1 indirect block, 1 pointed to by indirect block, 1 for the directory entry block and 1 to read the inode). If the first 638 files are deleted, the entries for . and .. are always defined, the directory will stay the same size. It will still take 14 disk reads to get the inode of the only file in the directory. Again HFS rides to our rescue to save us from this primitive environment.

Series 800 HP-UX directories come in two forms. Directories may have a maximum file name size of 14 characters or a maximum of 255. But, because of the large blocks used by HFS and because there is structure added to the directory entries, the problems that other UNIX's have are mostly overcome by HP-UX. While large full directories still require the file name to be found via a sequential search of the directory blocks, the larger blocks reduce the number of disk reads required. The internal directory entry structure permits HP-UX to only look at used entries instead of used and empty ones. This corrects the problem of large sparse directories.

With directories you obviously want to keep the number of files in them small, one or two blocks. If you do have a directory that contains hundreds of transient files, be sure to remove the files when they are no longer required. If all of the files in the transient directory do not get deleted and the directory requires more than one block, move the ones that remain. Either by hand or with a script, perform a mv operation on each file that remains after the massive purge. Move it to a different directory and then back, or to a different name in the same directory and then back to the original name. This will place all of the files in the first block of the directory. Just because HFS permits you to have 255 character file names doesn't mean you have to create them that large. System performance and personal sanity will be better served with reasonably sized file names. I would use the large file names option (newfs -L) when creating file systems. Not for the ability to have long names but because each directory entry only uses the number of bytes it requires. The short file name entries (newfs -S) are all 32 bytes long while the long entries only use the space required for the name plus overhead. Generally the long file name file systems will place more directory entries per directory block than the short file name systems.

Attempt at humor

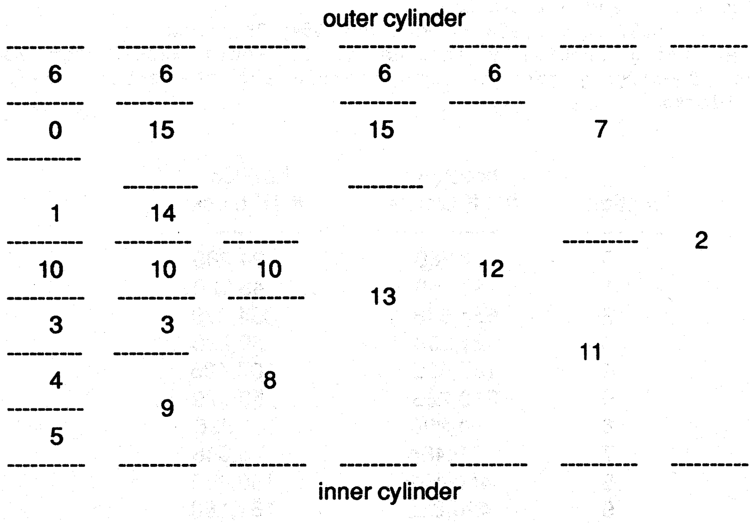
According to HP-UX Reference manual Volume 3, page 471, "You can tune a file system, but you can't tune a fish.". This is wrong! You can tune a fish if you know the correct scale and do it in the key of sea. Tuning a file system is what this presentation is about. I will leave fish tuning as a home work assignment.

Tuning

The commands I will be discussing are newfs and tunefs. Newfs is a front end command to mkfs and so the mkfs options will also be discussed.

File System Sections

When creating a file system, the first parameter to be determined is where on the disk you want to place it. Series 800 disks are divided into sections. There are up to 16 sections on a disk depending on it's type. The disk type also determines which sections overlap each other and the maximum number of non-overlapping sections. The file /etc./disktab describes the disk sectioning. An example of a disktab entry is the following for disks greater than 350M bytes and less than 2G bytes. The numbers are the section numbers. This is the number that is used in the file system pathname as the last number (following the s). /dev/dsk/c1d0s7 identifies section 7 of the disk. The table shows that sections 15 and 14 represent the same area of the disk as sections 0 and 1. And section 9 overlaps sections 4 and 5. Section 2 overlaps all of the other sections. Since the file systems on a disk can not overlap, if you decided that one of the file systems was going to be section 13 the other file systems would have to be sections 6 and 15. You could pick 6 and 0 but then you would be wasting disk space.


outer cylinder
6
6
6
6
0
15
15
7
1
14
2
10
10
10
12
13
3
3
11
4
8
9
5
inner cylinder

I find the figure from the "How HP-UX Works: Concepts for the System Administrator" book page 8-12 easier to visualize the disk sectioning concept.

For disks of this type, you could specify the entire disk to be used by one file system by specifying s2 as the section portion of the pathname. Or, you could have 7 file systems on the disk by specify s6, s0, s1 s10, s3, s4 and s5 as the file system sections.

If the disk was an hp2203 and you made file systems using sections 13, 6 and 0, as mentioned earlier, you would use

603,872 + 1,998 + 24,280 = 630150K blocks.

Specifying 13, 15 and 6 would use

603,872 + 1,998 + 48,560 = 654430K blocks.

Therefore, using section 0 instead of 15 would waste over 23 megabytes. Specifying just section 2 would use the entire disk, 654,948K blocks.

section	hp2203 # 1K blocks	hp7935 # 1K blocks
0	24,280	24,280
1	48,560	48,560
2	654,948	394,979
3	29,298	29,298
4	107,426	107,426
5	313,236	53,520
6	1,998	1,998
7	75,484	75,348
8	450,192	190,462
9	420,812	161,160
10	129,024	129,924
11	579,464	319,630
12	652,688	392,886
13	603,872	344,148
14	24,280	24,280
15	48,560	48,560

newfs

Now that we understand the disk sectioning we are ready to create our file system(s) on the disk. This section will explain how to do this using the newfs command. Only the optional arguments that when changed from their default values may affect system performance will be discussed.

newfs requires a disk section pathname and a disk type. The section pathname must be of the form /dev/rdsk/cxxd0syy where xx is the logical unit number of the disk and yy is the desired section on the disk. The disk type is defined in the /etc./disktab file. To determine the disk type of a given disk, enter the diskinfo command and look at the returned product id number. For example on the system I am using

```
diskinfo /dev/rdsk/c1d0s7
```

returns a product id of 2203. Therefore, my disk type is hp2203. The newfs command that uses default values for the optional arguments would be:

```
newfs /dev/rdsk/c1d0s7 hp2203
```

The optional arguments I will be discussing are b, f, c and m.

The Series 800 HFS permits us to define the block size to be used by the file system. In order to guarantee that no file will require triple indirect addressing within its inode, the minimum block size is defined to be 4096. The only other size we will consider is 8192. Blocks can be made as large as 64K, but that is not a realistic size for the environments we are discussing. A block is read or written when a program performs file I/O. The block is kept in memory until the space is required for other activity or the file is no longer being used. A larger block, for some types of files, will result in better system performance because there will be fewer disk accesses required to read the file's data. There are some cases that a larger block size hinders performance. This will be discussed later.

Fragment

As I mentioned earlier, the HFS places the last part of a file in a fraction of a block. This fraction is called a fragment. The concept of block fragments is included in the HFS in order to prevent the last block of every file on the system from wasting an average of 4K bytes. As a file grows, the data that makes up the last block of the file is placed in a fragment of a block. If the defined fragment size is 1K , and a file is between 8K and 9K bytes in size, the first 8K bytes will be placed in a block and the bytes left over will be placed in 1K bytes of a block being used for fragments. Once the file grows beyond 9K, the fragment will require another 1K. If within the fragment block the 1K following the current fragment is free, it will be allocated for the file. If the next 1K fragment is not available, a 2K fragment must be found in a different fragment block. The current 1K fragment is moved to the new 2K fragment and the new data added to the newly allocated area. This process is continued until the fragment fills an entire block. Fragment blocks are only used for the direct access blocks (the first 12). Once data blocks have to be accessed via indirect blocks, only whole blocks are allocated.

A test I ran on a new file system that had 1K fragments and 8K blocks pointed out that fragments are moved even when the file system is empty. I then tried a file system with 8K fragments and 8K blocks. And found a slight degradation in performance, My tests showed a performance improvement when the block size was 8K and the fragment size was 4K. It is surprising that there is a measurable difference since only the first 12 blocks worry about fragments.

Cylinder Group

The number of cylinders that make up a cylinder group is also a configurable parameter. The default value is 16 cylinders per group and the maximum value is 32. There is a small amount of overhead associated with each cylinder group to manage the blocks and inodes. If the number of cylinders per group is too small, this percent of overhead blocks to usable blocks becomes high and a significant portion of the disk is not available for user data. Also, more processing power may be required to manage many small groups instead of a few larger groups.

Free space

The -m option permits you to specify how much of the file system will not be available for normal use. The default is 10 percent. This means when your file system becomes 90 percent full, no one except the super user can allocate any more space to any file. This parameter is intended to prevent system performance from degrading. It reduces the time required to find free space when a file is expanding. By having the minimum free space at 10 percent, or larger, the system is suppose to run better because there is a greater locality of reference when accessing files. The disk doesn't have to search as far from where it currently is to find the next block the OS has requested.

tunefs

Once a file system is created, the command tunefs can be used to modify some more file system parameters. The two I will discuss are d and e (the percentage of free space, -m, may also be set by tunefs).

Rotational Delay

The -d argument specifies the time in milliseconds it takes to complete a data transfer and initiate another one. Using this information, the system can determine the optimal positioning of a file's blocks on the disk. If the blocks are placed correctly, then when they are read the data seek times will be minimal. As the disk platter is spinning, all of the blocks that have been placed on a given track would be read in one revolution of the platter instead of requiring about one revolution per block. This is a very sensitive parameter and should not be touched by the faint of heart.

Blocks per group

-e defines the maximum number of blocks per cylinder group that any one file is permitted to allocate. The default value is about 25 percent of the total block in the cylinder group. This is to prevent one large file from using all of the blocks in the group and forcing the other files within the same directory to be placed in other cylinder groups. This parameter tries to ensure locality of reference for most files with their inodes.

Tuning ATG (According To Glen)

If the files on your system are all small bin and text files and you use a data base that manages its own partition HFS is built for you. The default parameter settings were set with you in mind. Especially if your system is seldom used by more than a couple people at a time doing highly interactive work.

If, however, you have a heavily used system that has several large files. And, those files are a type of ISAM (index sequential access method) file, such as C-ISAM or TISAM. Or, if you use your own indexing scheme built into a flat file, read on.

One of the great philosophers of my generation has said many times;

"You can't always get what you want
But if you try some times
You just might find
You get what you need."

HFS is not what I want . Let's see if it is all I need.

File System Sections

The HFS does force us to pick 1 of up to 16 locations on the disk for our file systems. This isn't what I want . I would like to have more freedom. I hate external constraints. But, it is definitely all I need (especially if the Logical Volume Manager is used, but that's another paper).

My goal in life is to keep things as simple as possible. To that end, if resources are not a limiting factor, I would have one file system per disk. This simplifies performance tuning since I would now only have to worry about load balancing disks and not file systems on disks. One of the main reasons the HFS has cylinder groups is to reduce disk seek times by increasing the likelihood that successive disk access are physically near each other. By having multiple file systems on a disk, we are forcing the possible accesses to be far apart.

Unfortunately we, or at least I , don't live in a perfect world where the accountants give us all of the money we want for equipment. Therefore, in this imperfect world, we do have to place multiple file systems on some of our disks. Some files perform best when the file system is tuned one way and others when it is tuned another way. We obviously want to separate these files into different file systems. Bin files are generally accessed best on a file system that uses the default setting while large data files require different settings.

Or , if during the day time one set of files are primarily accessed and in the evening, when reports are run, a different set are accessed, it would make sense to separate them into two file systems. If they were kept in one file system, the blocks of both sets would be spread across the entire disk. If each set was in their own file system, then their blocks would be closer to each other.

If you have to use multiple file systems, try to put an infrequently accessed one (or two) on the same disk as one that is frequently accessed.

I like some things

For files that are opened and then the whole thing, or large chunks, are sucked into memory is one read, the HFS is what I need and want. The large blocks, rotational spacing, and inode placement algorithms are beautiful. Every millisecond of performance is accounted for while still being realistic.

Tuning Parameters

For large data bases that are randomly accessed, though, tuning is required.

-b

Large is a relative term. If your system has 100 megabytes of data that is accessed most often (80% of the time) and you can use 25 megabytes of memory for buffer cache, that is not large. 100M of data in this case isn't large because there is so much buffer cache that you can expect a high buffer cache hit to miss ratio and thus logically reduce the amount of data. In almost all cases like this it is best to use a 8K block size. If, however, your system only has 2 megabytes of buffer cache, very few of the disk buffers can be cached and the buffer cache hit ratio becomes small. In cases like this, a smaller buffer results in better performance.

-f

The fragment parameter only affects performance when files are being increased in size. It is possible with a 1K fragment and an 8K block to have to move the original 1K of data seven times before it is placed into an 8K block. This file data movement causes system overhead which degrades performance. Fragmentation is only used with the first 12 blocks of a file. After that , an entire block is allocated every time. I had expected this parameter to have little affect on performance on the building of a large data file. I was surprised. With an 8K block I had best performance when using a 4K fragment With a 4K block a fragment of 1K worked better than a 4K fragment. These results for a large data base are more interesting than important. However, for a file system that has many small transient files I believe an 8K block size and 4K fragment would work best.

-c

For large files, cylinder groups just get in the way. By setting -c to the maximum value, 32, and making the recommend modifications to -d and -e file accesses are a little faster. It is not significant and leaving it set to 16 probably won't make a noticeable difference on your system. Be sure to make the -d and -e changes. Just changing the -c to 32 actually caused my tests to run a little longer. The amount either way is small, but every little bit helps.

-d

-d 0 will cause the blocks of a file to be placed contiguously on the disk, if possible. For ISAM or ISAM like files, the blocks are not read sequentially. To perform a single record read, block 10 then 31024 then 392 then 51324 may have to be read. Using rotational delay to try to time the placement of block negatively affects performance because it increases the scattering of the file's blocks. -d 0 by itself, helps performance to some extent but by adding the -e
modification, large files are helped a lot.

-e

This option specifies the maximum number of blocks a file can use within a cylinder group. Remember the purpose of cylinder groups is to increase the locality of reference between a file's inode and the first several blocks of the file. For large files we want locality of reference to be as small as possible between the blocks of a file. I don't care how far from the inode the data is because it won't be accessed very often compared to how often the data is accessed. Therefore, set -e to the blocks per group value. If you don't want to look for that value (by doing a tunefs -v) just enter -e 99999. That is large enough. -e by itself will not keep the block of a file close to each other. You must also specify -d 0.

If you do not, the system will permit the file to use the entire cylinder group for one file but since the allocation scheme is skipping X number of blocks between allocations, when it gets to the end of one cylinder, instead of using free blocks earlier in the group, it will move on to the next cylinder group.

-m

For file systems that are used for large data files and tuned as I have suggested above, I would set -m to 1. Leave a little space for the super user to maneuver. The purpose of leaving 10 percent free space is to aid performance. For file systems that contain mainly small transient files, this may help while allocating files. But for file systems with a few large files, all this accomplishes is to waste 10 percent of your disk. The free space is not kept equally spread among the cylinder groups. depending on how the files in the files system are created and expanded, the free space may be all in one cylinder group or spread around in a few. The free space is not forced to be spread among the cylinder groups. It is not the amount of free space in a file system that affects performance but the distribution of the files. If we are suppose to not use 10% of our disks, then H.P. should charge us 10% less for the disks.

The -b, -d and -e options affect performance the most. The others, -m, -c (16 or 32) and -f for large file systems have minimal affect.

Also, be aware that when directories are created they are placed in the least full cylinder group. The files that go into that directory will then be place in the same cylinder group as the directory, if possible. Therefore, if a file system has 12 cylinder groups and you create 12 directories in it each containing one small file. You may be only using one percent to the disk, but you are requiring the system to span the entire disk in order to access the data.

A Bold Statement

File systems that are heavily used by multiple processes and are nearly full, generally do NOT adversely affect individual process performance or overall system performance if the files are overly fragmented. This is because the next block to be read is likely from a file other than the current one. Since the files are spread across the entire file system, the average seek distance would be half of the disk. If the file were not fragmented, the average seek distance would still be half of the disk. But, if

1) the files are spread across the entire file system and the system is less than two thirds full, or
2) the file system is lightly used by multiple processes or
3) the file system is used by a single process such as a report generator

performance is adversely affected by file and file system fragmentation . All of these situations cause the average seek time for each disk I/O to be longer than if the files on the file system were contiguous. One way to correct this situation is to create a new file system and copy all of the files, one at a time, from the fragmented file system to the new one. A less attractive way is to backup the file of a system to tape, recreate the file system and then restore the files. This is a scary method because the files must be deleted from the disk and only be on tape. If you use this method be sure to have multiple good backups (color me *paranoid*). A better way is to use a utility to unfragment the file system in place. This is better but to my knowledge not yet available on the HP-UX. There is at least one company working on porting a file system compression program to the HP-UX, but at this time it is still futureware.

Wish List

I would like to be able to specify a cylinder group of 64 or (gasp) 128 cylinders instead of a maximum of 32. Or, even better, would be to say that the entire file system was one cylinder group. This would reduce the system overhead of jumping between cylinder groups as well as give me back some disk space (I know, picky, picky). I would not use this feature for all file systems, but it may help in some cases.

I would like to be able to turn the fragment block feature off. If I say -f 0 while doing a newfs, let that mean to always allocate full blocks to the file. Never add the overhead of moving fragments just to save a little disk space. I am still researching the strange results I saw in my testing. When I have a block size of 8K, a 4K fragments is optimal . I had thought an 8K would be best. An 8K fragment actually gave me slightly worse performance,. When I have a block size of 4K, a 1K fragment is best. Perhaps at Interex I will have this anomaly explained.

I would like to be able to set the block size to 1K or 2K. For ISAM files that perform I/O in 1K blocks, having larger file system block sizes adds needless overhead.

Managing ISAM Files

The rest of this paper concerns managing ISAM files. If you do not use ISAMs or are not responsible for maintaining your data base, it probably isn't for you.

In general files on the HP-UX system, as is true with most operating systems, should be kept as small as possible. This is not always something you have control over, but as you will see in the next section, you do have some options with most of your large data base files.

If you do not use a data base that manages its own partitions, your large files are likely to be ISAM files. ISAM is an acronym for indexed sequential access method. Some of you are using a version of this file type created by Texas Instruments called TISAM and the rest are likely using a version created by Informix called C-ISAM or a derivative of it. The internal structures of the two versions are different but they use the same concepts.

Basics

Each uses two UNIX files to represent one ISAM file. There is the data, or dat, file and the index, or idx, file. The files have the same name and are distinguished by the UNIX extensions of .dat and .idx. Some implementations of the C-ISAM version do not place the .dat after the data file name. Only the .idx is added to the index file to distinguish from and associate it with the data file.

Data File

The data files are just a byte stream like any other HP-UX file. TISAM files do have a lot of internal structure per record but the records are organized in the file as a byte stream. The records are not in ISAM defined blocks. Record X is physically followed by record X+1. There are no unused bytes within the file until records are removed from it.

The version of C-ISAM file you are likely using does not provide any form of compression. Newer versions do, but we will concentrate on the version you are most likely using. All of the bytes within the file are bytes your programs have written to the file except one per record. Each record written to the file has a line feed character, hex a, appended to it. Therefore, the total size of a data file is the record size plus one times the number of records in it. This assumes there have been no records deleted from the file.

TISAM data files have a lot of structure added to each record. Some of these overhead bytes are to aid in TISAM's implementation of data integrity. Most of them are used to implement TISAM's same character compression algorithm. Most records will have consecutive bytes with the same value. Usually the bytes are blanks but they may be any value. By using a compression scheme, a stream of 20 blanks, for example, can be internally stored using only 3 bytes. Most data file records have enough instances of consecutive same bytes that even when the overhead bytes are added to each record to implement compression, the overall size of the record becomes smaller. Most data files become significantly smaller. I have seen some that are one sixth the size of what they would be uncompressed.

Index File

An index file contains structures that permit programs to store and retrieve records from the data file using the content of predefined fields of the records. These predefined fields are called keys. Other files store and retrieve data by the position of the data within the file. The program must know that the data it wants starts at byte X and is Y bytes long. ISAM files use keys to permit programs to specify they want the record with a social security field of 123-45-6789.

There are also internal structures to keep track of the locations in the data file that have had records deleted. This space is then available for new records to be added. The index file may also have locations freed up when records are deleted from the data file since the corresponding keys must also be deleted. Therefore, there is also an internal structure for free index file locations.

Unlike the data file, the index file is not a byte stream. The file is divided into fixed length blocks or nodes. The block size is usually 1K. Thus, the free locations within the index file stored in the internal structure only handles blocks that are totally empty. The normal state of index blocks that contain keys is that there are unused bytes in them. Rarely are the blocks in an index file full.

If most of the index blocks of a file are only partially full, unnecessary disk reads may be required in order to read and write data file records. Even when the index blocks are completely full, at least one and usually several disk operations must be performed before the data file is accessed. The additional disk accesses are required to implement the ability to access data by content instead of location. Your goal is to eliminate all unnecessary disk activity. The last section explains how to do this.

Required I/O

First, however, let's determine what disk activity is necessary. If the keys of a file are defined to not use the key compression features, it is possible to calculate exactly the number of necessary disk reads that are required to transverse the index file structure. If compression is used, only educated guesses can be made. For this discussion we will assume the keys are not compressed.

The index file structure is called a tree. The tree is made up of nodes. All accesses to a tree start by reading the same node called the root. If all of the keys cannot fit within the root node, the root node subdivides the keys into other nodes. The most keys that could fit into the root node is the block size minus a few overhead bytes divided by the internal key size plus some bytes of overhead. The additional key overhead varies from 4 to 11 bytes depending on the key attributes . If the keys cannot fit in the subdivided nodes, they are defined to be a branch node and that further subdivide the keys into more nodes. This subdividing continues until a level is reached that contains enough nodes to hold all of the keys. Nodes at this level are called leaf nodes. The keys within the leaf node point to the actual information within the data file.

For example, let's assume the maximum number of keys per node is 100. This would be the case for keys with a defined size of 6 bytes or fewer depending on the key attributes. If the data file contains 100 or fewer records, then all of the keys could fit within the root node. The tree in this case would only be 1 level and thus only one disk read is required to determine where to read the record from the data file. If there are 101 records, then two nodes are required in the tree. The root node would point to two other nodes. One of the them would contain 50 keys and the other 51. Note that the keys are split between the two nodes. One doesn't contain 100 keys and the other just 1. Since the root node can contain 100 branch node pointers and each branch node can contain 100 keys, the maximum number of records that can be identified using a two level tree is 100 times 100 or 10,000. But, this is not a realistic number. Since during normal operations the leaf nodes are not completely full, a more accurate number would result by assuming that all nodes under the root node are about 75% full. So a two level tree would hold about 7,500 pointers (100 * 75). A three level tree could hold pointers to 562,500 (100 * 75 * 75) records.

The above example was for a small key. If the key was about 30 bytes long, it would take six levels to manage a half million records! That adds a lot of drag on the performance of your applications as well as the performance of the entire system.

The tree structures require at least one additional disk read before the data can be read. This is a fact of life when using ISAM files. But all unnecessary disk operations should be eliminated. The key trees should be maintained at the minimum number of levels possible. Both the data and index files should be kept as small as possible to prevent the problems all UNIX files have when they become large. When records are deleted from the data file, the file remains the same size. The freed space is internally managed. To eliminate the internal file fragmentation and reduce the size of the file, it must be rebuilt.

Similarly, deleting keys from a tree does not reduce the number of levels in the tree. If before a large number of records are purged from the file, it has a tree that is six levels deep, even if all of the keys are removed from the tree, it will remain six levels deep. The only way to decrease the number of levels is to rebuild the index tree after the keys have been removed.

File Rebuilding

Most systems have three methods available to rebuild ISAM files. The first is to copy the records from the ISAM to a flat file, delete the ISAM, create and empty ISAM and copy the flat file to new ISAM. This operation can be sped up by creating the new ISAM and copying directly to it from the old ISAM. Either way requires enough disk space to keep the data online twice. This method is also the slowest. For files with many records, several keys or very large keys, this operation may take several days. If the index or data file is corrupted, this method will likely not work.

BCHECK and VFISAM

To correct corrupted files, C-ISAM systems provide a utility called BCHECK and TISAM systems provide VFISAM. These utilities can be used to rebuild index trees to reduce the number of levels of the key trees and eliminate the internal fragmentation of the index file. This is done by replacing the existing index file with a newly created one and executing the command. This is the best way to rebuild the trees. If the existing index file is used, the utility must first delete the trees. Then while rebuilding the trees, it will reuse the nodes in the order they appear on the free node structure. The resulting index file will not be smaller and the trees will be spread through out the file. VFISAM can also be used to compress the data file. It does not reduce the size of the file, but all of the data will be contiguous from the beginning of the file and the free space moved to the end. Since these utilities use the ISAM provided add index command, the resulting key nodes are not completely full. The end size of the index file will be about the same as the first method. Both of these utilities are faster than the first method but they can still take a long time. In one case, we had a TISAM file with 10.5 million records and two keys. It took VFISAM 9 days to rebuild the index file!

ISAMATION

The third method is a suite of utilities and scripts created by

Computer Solutions Inc. to compress and correct corrupted ISAM files in place and very quickly. It is generally ten times faster than BCHECK and VFISAM (it took 19 hours to rebuild the 10.5 million record file). Both the data and index files are also made as small as they can be.

ISAM files may become corrupted due to problems with hardware, software or poltergeists. When this occurs, often very little of the file's data is actually lost. It is still on disk but you cannot access it . VFISAM and BCHECK will be able to rebuild most of these files but since it uses the ISAM add index command to create the indexes, it is slow. Also, if the data file contains parity errors, these utilities will terminate. ISAMATION uses its own sorting algorithms to greatly speed up the building of indexes. If it encounters a parity error in the data file, it will by-pass the bad block and continue processing.

Conclusion

Many times the data file is not corrupted at all. It is the index file that is no longer correct. In these cases, no data is lost and all that is involved is time to rebuild the index. You don't have to have files with millions of records in them to cause your system to be down for a day if they have to be rebuilt. Files with many keys or with large keys and only a few hundred thousand records will also take an extremely long time to rebuild.

While rebuilding a corrupted ISAM file usually has to be performed, "right now", performing data base maintenance operations can be scheduled. Keeping ISAM files compressed not only saves disk space but usually helps overall system performance. Most shops reorganize their files after their yearly archive and purge takes place. This is a good idea. But, what has surprised me is how fragmented ISAM files can get with just normal use. And, how much file system space can be recovered by regularly reorganizing them.

This concludes my sermon.

PAPER NUMBER: 4015

TITLE: Real-World UNIX for Open System Enterprise Document Management

PRESENTER:

David Weinberger
Interleaf, Inc.
Prospect Place
9 Hillside Avenue
Waltham, MA 02154
617-290-0710

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

PAPER NUMBER: 4016

TITLE: Overview of DME

PRESENTER:

Andrew McCasker
Systems Center
1800 Alexander Bell Drive
Reston, VA 22091
703-264-8000

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

PAPER NUMBER:	4017
TITLE:	X Application Performance on Client/Server Systems
PRESENTER:	Ken Oliver Hewlett-Packard Co. c/o Ella Washington 19091 Pruneridge Ave. M/S 46LG Cupertino, CA 95014 408-447-1053

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

UNIX: The Universal Gateway For Cross-Platform Communications
Paper No. 4019

Gwen Peterson
Vice President, Marketing
Clarity Software
2700 Garcia Avenue, Mountain View CA 94043
(415) 691-0320

Abstract

With the built-in networking capabilities of UNIX, workstation users have been dependent on electronic mail for communications; with the growth of PC LANs, PC users are realizing the potential for this technology as well. But a gulf exists between the two islands of computing: to communicate across it users must either limit themselves to ASCII text or force-fit a single e-mail system on all users.

A software product exists for the HP 9000 that bridges this gap -- not only allowing the sending of robust compound documents between PCs, Macintoshes, and UNIX users, but automatically converting documents between users according to their profiles of preferred application formats. Clarity Rapport products allow HP workstations to become communications hubs for cross-platform electronic mail and faxing.

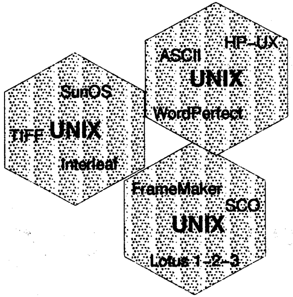
UNIX: The Universal Gateway
for
Cross–Platform Communication

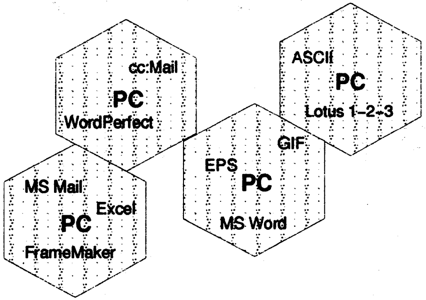
Gwen Peterson
Clarity Software

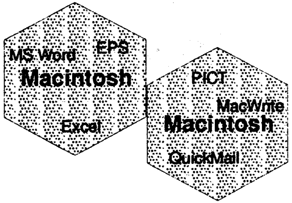
September 22, 1993

Agenda

- The Problem
- Interim Solutions
- Optimal Solution

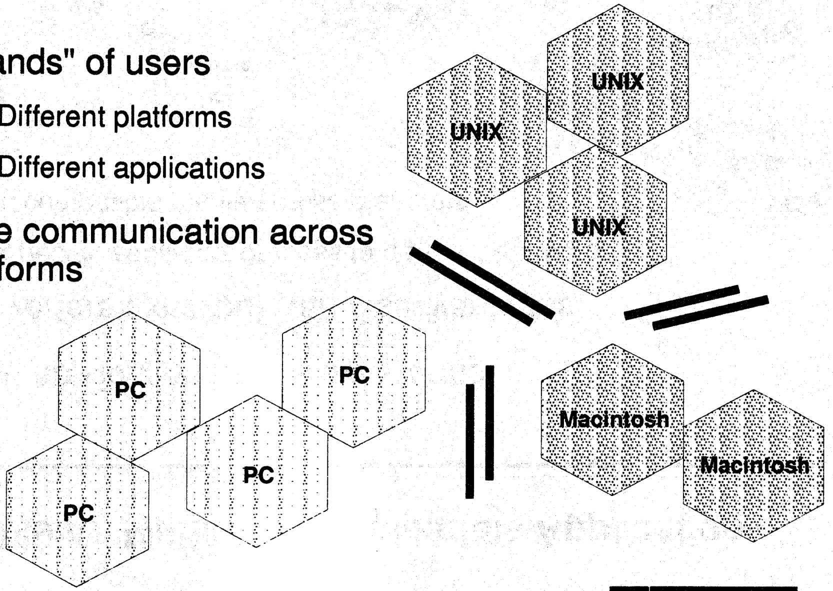

SunOS
TIFF UNIX
Interleaf
ASCII
HP-UX
UNIX
WordPerfect
FrameMaker
SCO
UNIX
Lotus 1-2-3


cc:Mail
PC
WordPerfect
ASCII
PC
Lotus 1-2-3
GIF
EPS
PC
MS Word
MS Mail
Excel
PC
FrameMaker


MS Word
EPS
Macintosh
Excel
PICT
MacWrite
Macintosh
QuickMail

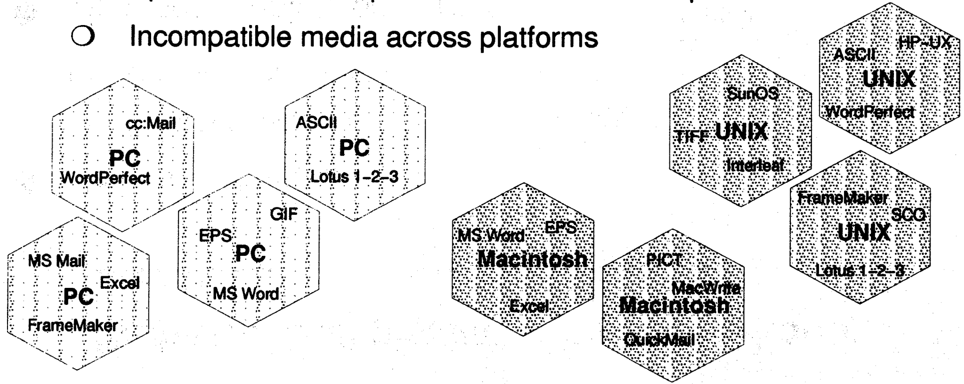
Problem: Multiple, Incompatible Platforms

- "Islands" of users
  - ○ Different platforms
  - ○ Different applications
- Little communication across platforms


UNIX
UNIX
UNIX
PC
PC
PC
PC
Macintosh
Macintosh

Problem: Multiple, Incompatible Applications

- Each application has its own format
- Converters exist; but with disadvantages:
  - ○ Special, additional process to do – not transparent
  - ○ Incompatible media across platforms



cc:Mail
PC
WordPerfect

ASCII
PC
Lotus 1-2-3

MS Mail
Excel
PC
FrameMaker

GIF
EPS
PC
MS Word

MS Word
EPS
Macintosh
Excel

PICT
MacWrite
Macintosh
QuickMail

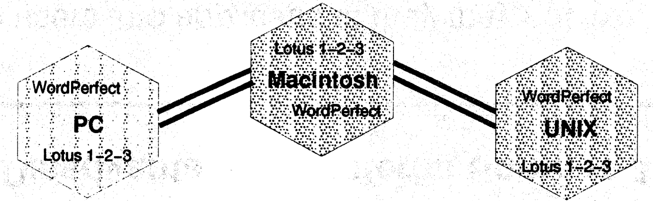
SunOS
TIFF UNIX
Interleaf

ASCII
HP-UX
UNIX
WordPerfect

FrameMaker
SCO
UNIX
Lotus 1-2-3

Interim Solution 1: Same Application Across Platforms

- Advantages:
  - ○ No or little file conversions needed
  - ○ Reduced training and support load for end users
- Examples:
  - ○ cc:Mail, Lotus 1–2–3 from Lotus Development
  - ○ WordPerfect from WordPerfect


WordPerfect
PC
Lotus 1–2–3
Lotus 1–2–3
Macintosh
WordPerfect
WordPerfect
UNIX
Lotus 1–2–3

Disadvantages of Interim Solution 1

- Force users into applications they may not like
  - ○ A "standard" on one platform may be unfamiliar or poorly implemented on another platform
  - ○ Users may have already learned other applications and have files in other formats – lots of up–front training and conversions required to make them switch
- File formats are not always the same
- Does not address the media issue
- Does not address interoperability needs outside the organization

Interim Solution 2: Universal File Viewer

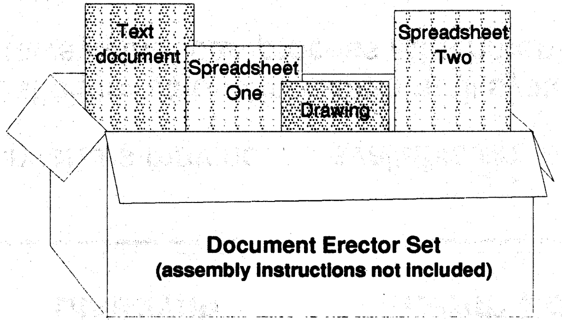
- Everyone can see what you have created
- Compound documents look [almost] the same to everyone
- Examples:
  - Adobe's Acrobat
  - No Hands' Common Ground
  - Farralon Computing's Replica
- Can get similar results via electronic fax

Disadvantages of Interim Solution 2

- Recipients cannot edit received documents (though that's good for distribution of manuals, etc.)
- Expensive; buy new software just for this purpose
- Everyone has to have the same viewer or it doesn't work

Interim Solution 3: E-mail Attachments

- Send whatever files you work with; let each recipient worry about converting it
- Files come as independent attachments


Text document
Spreadsheet One
Drawing
Spreadsheet Two
Document Erector Set
(assembly instructions not included)

Disadvantages of Interim Solution 3

- Converters may not be available on receiving end
- Must either all use the same mail system or have gateways that can process attachments correctly
- Attachments have no inherent relationship to one another; sequence is unclear

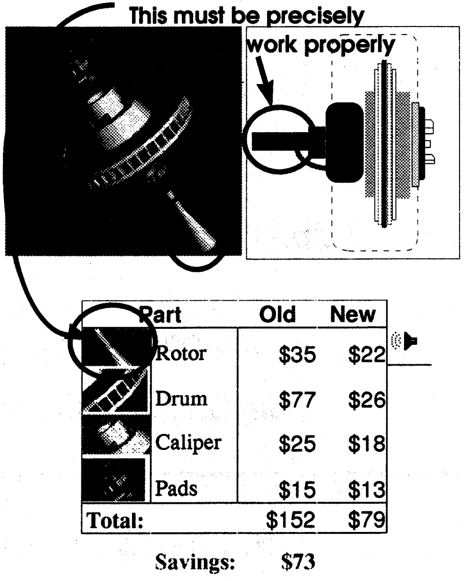
"Like getting a document as a collection of separate paragraphs, with no idea how they go together."

The Optimum Solution: Wish List

- Everyone can use the applications they prefer
- No extra steps are needed to accommodate conversions to other applications
- Multimedia documents retain information regarding all document elements
    - All elements received in place, or
    - Document is annotated to reflect where attachments belong
- Keep costs down
- Don't have to have a special environment on receiving end

Optimal Document Composition

- Integrates information from popular productivity applications
  - ○ Word processing
  - ○ Spreadsheets
  - ○ Graphics
  - ○ Vertical applications
- All applications are embedded as objects
  - ○ Editable within the document
  - ○ Intermix any way you want



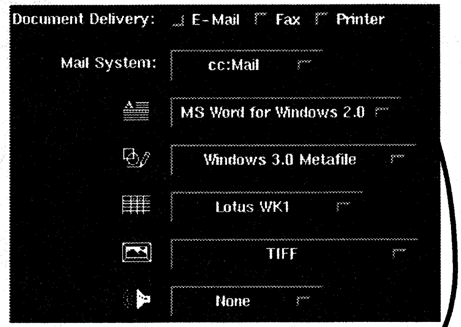
This must be precisely
work properly

Part	Old	New
Rotor	$35	$22
Drum	$77	$26
Caliper	$25	$18
Pads	$15	$13
Total:	$152	$79

Savings: $73

Getting The Message Across Optimally

- Conversions are automatic through electronic mail
  - Each person has an individual Address Book profile
  - Incoming conversions are also automatic
- Files received are formatted and editable in the applications of each user's choice
- Mail is formatted as each mail system expects it



Document Delivery: ☑ E-Mail ☐ Fax ☐ Printer

Mail System: cc:Mail

MS Word for Windows 2.0

Windows 3.0 Metafile

Lotus WK1

TIFF

None

- Message goes out in a form usable by each recipient

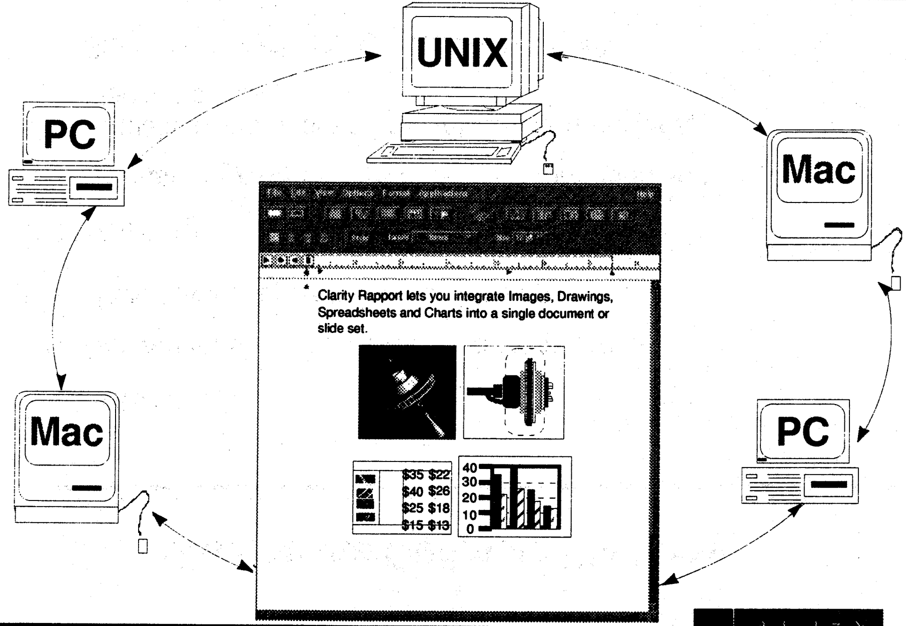
Receiving Cross-Platform Messages

- No special software required by recipients
  - Use whatever applications they want
  - Use whatever mail system they have
  - No additional cost
- Files received are formatted and editable in the applications of each user's choice
- Changes can be made and the files mailed back; incoming conversions are done automatically as well

UNIX as the Optimum Platform

- Multitasking
  - Conversions done as a background process
  - Have several applications active at once
- Networking
  - Operating system is designed for information sharing
  - Virtually every system has network connections, electronic mail
  - Standard e–mail transports: SMTP, X.400

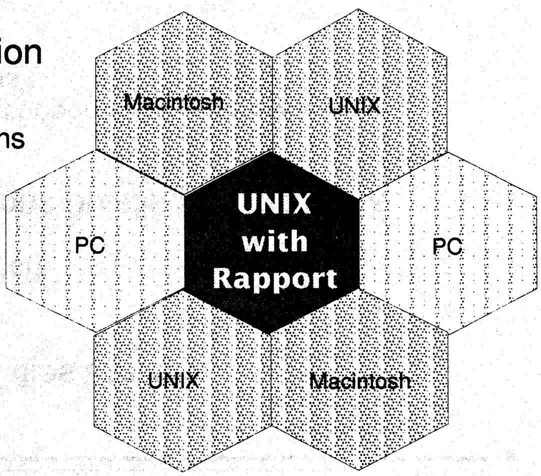
The Optimal Solution: Transparency of Platforms and Applications


UNIX
PC
Mac
Mac
PC
$35 $22
$40 $26
$25 $18
$15 $13
40
30
20
10
0

Clarity Rapport lets you integrate Images, Drawings, Spreadsheets and Charts into a single document or slide set.

Example of UNIX-Based Optimal Solution: Clarity Rapport

- Object-based composition
  - ○ Use Rapport's tools or your preferred applications
  - ○ Inserted files converted automatically
  - ○ Everything "live" in a single window
- Cross-Platform E-Mail
  - ○ Automatic conversions based on user profile
  - ○ No special software needed by recipients


Macintosh
UNIX
PC
UNIX with Rapport
PC
UNIX
Macintosh

It Can Be Done!

- Underlying UNIX capabilities are key
  - ○ Multitasking
  - ○ Inherent networking support
- Well–designed applications building on the UNIX foundation
  - ○ Inherent recognition and support of heterogeneous platforms and applications
  - ○ Use of standards
  - ○ True object–based architecture

Paper Number 4020

Analysis of a Mainframe Replacement Project

by Gary Gagliardi

115 NE 100th Street
Seattle, WA 98125
(206) 522-0055

How can you have predictable success in moving from your mainframe? Predictable results are possible only with experience. By definition, experience at the beginning of a change like this is limited. Around the world, companies are just beginning to replace mainframes and they are learning about the process as they go. They are discovering that the process is difficult and full of pitfalls.

After working on a large number of successful large-scale computer downsizing projects—and after analyzing a number of other projects that have been less than successful—a pattern can be detected in the successful projects. Whenever the elements of this pattern are in place, the move from the mainframe takes place smoothly. When elements are missing, problems inevitably follow.

The Phased Rollout and Risk

There are a number of risks associated with computer downsizing. The best way to reduce risk is to use a phased approach in the downsizing project.

In a phased approach, each step forward should be made tentatively. It is a test of technology and products. You must put enough planning and resources into that test to give it an opportunity to succeed, but you cannot afford to pour an endless amount of resources into something that clearly doesn't work.

The idea of a phased implementation is to take the smaller, less expensive steps first to test the ground before making larger, more costly decisions. Each of the basic phases represents an increased level of confidence and investment. It also represents an increased amount of knowledge regarding the realities of making the transition from old, well-understood systems to newer ones that are not as well understood.

The five basic phases in a mainframe replacement project are:

The Information Gathering Phase where standards are selected and the information strategy is set.

The Project Planning Phase where products are selected, a team assembled, and the plan for the Proof-of-Concept site is developed.

The Proof-of-Concept Phase where the first mainframe application is replaced at one site.

The Multi-Site Rollout Phase where the first mainframe application is replaced at all sites within the business unit.

The Gradual Expansion Phase where the other applications are targeted to be moved off the mainframe in greater numbers and the project is expanded to reach out to other business units that are sharing the mainframe.

At each phase, you answer a specific question about the feasibility of the project before moving on to the next phase. In going through the process, you deal with the long- and short-term risks associated with moving from the mainframe. Each of the risk areas—long- and short-term technological, operational, economic, and political—must be dealt with at the appropriate phase of the project. It is either impractical, too expensive, or simply impossible to deal with these risks at a different phase.

The entire purpose of the phased approach is to deal with the various types of risks at the most appropriate and cost-effective point. You don't want to fool yourself into thinking you have navigated a certain area of risk early in the process when you really can't deal with it fully until much later. But neither do you want to confront a certain type of risk later in the process when you could have handled it less expensively much earlier. You can't forget any of the risks at any phase, but each phase focuses on a particular set of risks.

Ideally, we would like to eliminate all risk in the Information Gathering Phase. However, this is impossible, and companies that try to eliminate all risks at that phase create a series of problems for themselves because they are attempting the impossible. We discuss this problem in more detail in the next section entitled "Common Mistakes in Downsizing."

The Information Gathering Phase deals with long-term technological risk and sets the stage for long-term political issues. These long-term issues require the involvement, support, and understanding of the top management. In this phase, you should determine which technological standards are the most likely to make the project a long-term success.

*The Project Planning Phase deals with **short-term technological and political risks.*** During this phase, you first establish that the chosen products are already working using the standards you have targeted. You then get consensus from internal users on the need for change. A small group of those users want the change to such a degree that they are willing to be the "guinea pigs" for the Proof-of-Concept Phase. Other users only have to commit to the degree that they are willing to change after the technology has been proven.

*The Proof-of-Concept Phase deals with **short-term operational and economic risks.*** At the end of the implementation of one application at one site, you should know if the chosen hardware, software, and project team can integrate the new technology into your operations. You should also know the true cost of installing the new technology.

*The Multi-Site Rollout Phase addresses many of the **long-term operational and economic risks.*** By the time you have rolled out the first application to a number of sites, you should be able to discontinue supporting that particular application on the mainframe and start maintaining it in the new environment. At this time, you should see distinct economic and business advantages arising from the project.

*The Gradual Expansion Phase is necessary to deal with the **total long-term economic risk** and with the **long-term political risk.*** Unless you can move all applications off of the mainframe, allowing you to turn it off completely, you are never going to get the cost reductions you need. You also have to give the organization as a whole a technology direction it can utilize to reduce costs everywhere. If you don't, your "local" solution is never going to be politically successful. This is especially true when the mainframe is shared with other business units in the corporation.

Success at each phase is fairly easy to define. This is especially true regarding the "cost" issues: what you expect to pay and how long you expect the process to take. If a given phase runs dramatically over those targets, serious questions are naturally raised about the viability of going on to the next phase.

Before going from one phase to another, you need to make sure that you have answered the appropriate questions at that point in the process. If you haven't found satisfactory answers, you must be willing to repeat the stage or even fall back a step in the process to an earlier phase. A picture of this decision-making process is shown in Figure 6-3.

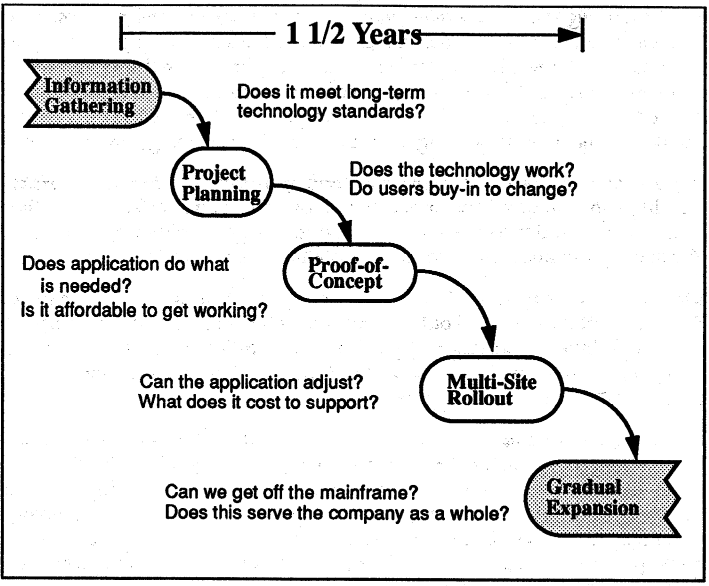

1 1/2 Years
Information Gathering
Does it meet long-term technology standards?
Project Planning
Does the technology work? Do users buy-in to change?
Does application do what is needed? Is it affordable to get working?
Proof-of-Concept
Can the application adjust? What does it cost to support?
Multi-Site Rollout
Can we get off the mainframe? Does this serve the company as a whole?
Gradual Expansion

Figure 1. PHASES IN A MAINFRAME REPLACEMENT PROJECT

Each phase acts as a test answering a basic question about the solution before you go on to the next phase.

Downsizing Phases

These Five Phases of Downsizing can be described in terms of:

- the people involved,
- the main purpose of their activity,
- the problems encountered,
- and the amount of time they take.

The Information Gathering Phase

During this phase, you are seeking a clear picture of what you are attempting to accomplish by replacing the mainframe. In other words, the purpose of this stage is to articulate exactly why you are downsizing.

One way to articulate these goals is in measurable economic and operational terms. For example, you may want to cut the cost of transaction processing by a certain percentage, or speed up a certain type of processing, performing it in days instead of weeks, or give people access to a type of information they have been unable to get in the past. These goals serve as the benchmark for the rest of the project. One of the purposes of this stage is to identify goals that are aggressive but reasonable.

From a longer term perspective, however, these economic and operational goals must be matched by an information strategy. This means identifying the standards that the organization needs to adopt in order to create a more connected and supportable base for corporate applications. The IBM mainframe was the old standard. This phase must articulate for the organization the required elements of the new standard. This standard is a long-term direction for the organization as a whole, not what is best for a particular part of the organization at a particular time. These standards are the foundation. They should be designed so that you build on it in the years to come. Without these standards clearly in place, good decisions about product selection are nearly impossible.

The common activities in this phase include:

1. **Involvement of senior management in review of the potential risks and rewards of mainframe replacement.**

The initiative to replace the mainframe can start anywhere within the organization, but the project cannot be said to truly begin until senior management becomes directly involved. This buy-in must be more than the authorization of a budget. The CEO and CFO of the organization are the only ones who can truly set the long-term strategic direction of the organization. This cannot be done within the MIS department. The MIS department can only attempt to fulfill the organization's requirements once this direction has been set.

This project starts as an educational process where senior management learns about the possibilities and problems of mainframe replacement. Without the knowledgable support and direct participation of senior management in setting these goals, moving from the mainframe has proven to be politically impossible.

2. **Appointing an information gathering team and giving them deadlines.**

This team's job is to articulate the long-term goals of the organization in mainframe replacement and to organize the search for possible ways to approach those goals.

This team should involve people from both operations and MIS. In some cases, the direct participation of senior management may extend to membership on this team or at least some direct oversight of its meetings. The initial job of this team is to articulate the information strategy. This teams provides the manpower to perform the remaining tasks at this phase.

This team needs a deadline. Without a deadline, the task of gathering information can expand beyond any reasonable scope.

3. Defining the goals and scope of the mainframe project in a Request for Proposal (RFP).

This proposal should not be the detailed specification of a solution, but a high-level statement of goals and standards for the future and a request for information from the market about how those goals might be met. This document should focus only on the "critical success factors" that define at a high-level whether or not this project will be a success. At this stage, a ten-page document is more useful and less expensive than a one hundred-page document. It should state the information management goals of the organization in way that are relevant and approved by senior management and in a way which can be understood and appreciated by outside organizations.

The challenge at this stage is to avoid getting bogged down in a huge amount of detail about the applications being replaced. The most common mistake is to create a large list of desired functionality without defining their critical qualities or effect on high-level goals. Given a relatively few measurable, prioritized features, you can get the information you need at this stage in the process.

The critical success factors parallel the risks associated with mainframe replacement. There are critical technological factors that are associated with long-term standards that reduce your maintenance cost over time. There are critical operational factors associated with the performance of the system and the cycle time of transactions. There are critical economic factors that are associated with the cost and time of installing the system and with how quickly you see a return on your investment and with how you measure it. Finally, There are critical political factors regarding who has to be satisfied with the system and who outside the company its implementation affects.

4. Distributing this high-level RFP to likely candidates for involvement in the project.

At this stage, the organization's attention turns to the marketplace to ask those in the marketplace for their input about how to best satisfy the organization's articulated long-term desires. The market input is absolutely critical at this point. Setting idealized goals that cannot be realized given current technology is a waste of time.

The information gathering team should have identified those organizations with the most experience and knowledge about the mainframe replacement process. These organizations should include Big Six firms, system integrators, system integration divisions of hardware companies, and application providers. All of these organizations should have experience in mainframe replacement projects. The team should distribute their RFP to the identified organizations, and the responses should suggest specific routes to addressing your long-term needs for mainframe replacement.

5. **Analyzing their responses to create a "short-list" of possible partners and technologies.**

As written responses are received, the respondents and their proposals need to be evaluated in light of how likely they are to fulfill the needs of the organization. Proposals which are simply pitching products with little understanding of your organization's needs or the process by which those needs can be met over the long term can be eliminated. The organizations that provide the more credible proposals should be selected for further interview.

6. **Interviewing candidates to determine their value as partners in the project.**

These interviews too should involve senior management. At this stage, the organization is looking for partners it can work with. This is determined as much as anything by the people involved and the "fit" between different organizational cultures. Before going on to the next stage, you must identify the organizations your company can work with and the people your company can trust. It is not necessary that you select the best of those people at this stage, but until you are comfortable that such companies exist, going on to the next phase in the process makes little sense.

Avoid attempting to answer all possible questions at this phase before proceeding forward. This simply cannot be done. Each phase can provide only a certain amount of the information you need. You can only get better information by moving forward in the process. If your project finds itself stalled, it is simply because you are trying to answer questions at one phase that can only be answered at a later phase. The best you can do at this phase is to develop reasonable economic and business goals and identify the standards you want to support in the long term.

This phase continues until you feel comfortable that your organization understands what is realistic in terms of economics, operational change, and technological standards. Ideally, you have also uncovered all potential vendor candidates to help you in this process. You do not have to pick your suppliers at this time, but you have to know who the candidates are and their relative positions in the market.

The Project Planning Phase

During this phase, management makes the decisions necessary to go forward with the Proof-of-Concept implementation. The goals of this phase are: to create a team to manage the project; to identify the application that should be replaced first and the site which will be the Proof-of-Concept site; to pick specific products from specific vendors; to put together a comprehensive project plan and timetable; and to get your internal users to buy into the plan.

This phase requires from two to three months given the availability of experienced people and a good planning guide. The activities in the process are:

1. **Selecting a core implementation team for the project.**

The first problem at this phase is finding a good team of people. You need an internal team and team management but you also need to out-source the project management supervision to get an objective, outside view of how well your internal people are doing. One of the common mistakes is relying too heavily on internal people who are inexperienced at this particular type of project. Projects are much more successful when they include independent third party service companies who have more experience at managing these types of projects. These third party companies are also much better positioned to deal with the short-term and long-term political issues involved in the project. Experience has made it clear that these external people are better at getting buy-in from internal users and much, much better at handling the later stages of the project where you are trying to get buy-in from the organization as a whole.

2. **Identifying a target application to move off the mainframe.**

The first job of this team is to identify a target application for moving off of the mainframe. There are several criteria for picking this application. The right application might be the one that:

- is the most costly to support,
- is the most "broken" in terms of fulfilling company objectives,
- has the greatest effect on customer satisfaction,
- is the most generally used application,
- is the least risky to replace.

In reality, most companies start with either order-processing applications or general ledger applications. Both of these applications are a primary use of the mainframe today. Order-processing is chosen usually because of issues relating to customer satisfaction and cost of support. The general ledger is chosen because it seems less risky to support and because of the need to get consolidation-type information together more quickly.

3. **Identifying a willing Proof-of-Concept site.**

The next step is to find a site willing to work as a Proof-of-Concept test. This site has to be motivated toward change for their own reasons so that they are willing to contribute the time and resources this initial effort requires.

4. Identifying the products and vendors team you need.

After picking your team and targeting an application and a site, it is time to make a decision about the vendors with whom you want to work. You start with a short list of vendors who can supply the type of technological standards the organization decided upon in the Information Gathering Phase. From that short list the team must now pick the most likely candidates.

The short list should already consist only of companies who can meet your long-term requirements for standards and who have a good idea of how to make this project successful.

Beware of selecting products on the basis of demonstrations alone at this stage. Demonstrations are extremely misleading. In them, every vendor shows their unique, exciting features, but those "exciting" features are never critical to the success of the project. The real issues of performance and suitability for your needs are not demonstrable. Too many companies have learned, much to their chagrin, that a great demonstration does not even guaranty that the product works.

The best vendors are those with the *most* proven successes in projects similar in size and complexity to your own. Every vendor can come up with one or two reference sites, but surprisingly few vendors can offer meaningful numbers of mainframe replacement installations in larger, credible companies. Certainly, the broader the base of experience and the higher the quality of users, the more confident you can be that the vendors can deal with the unique situations likely to be encountered in your own installation.

If the "real world installations" test still leaves you with a short list, the next decision point should be the quality of the vendors' installation plan and your impression of their ability to work with your team.

Seldom are more expensive tie-breakers needed to pick a product, but if necessary, a workable objective measurement at this phase is a relative performance test of different vendors' solutions. This process is more expensive and, perhaps surprisingly, less indicative of success than real world installation experience or the quality of the installation plan and people. In going through this step, you may actually lose the participation of some of the best companies because it raises their cost of sales to unacceptable levels. In any case, a comparative performance test is not a "real world" test. That can only be done affordably in the next phase. This is a overly simplified version of the real world that measures each vendors' transaction throughput with a standard database and platform that the various contending vendors can agree upon. In general, low volumes of throughput indicate higher costs for the system.

5. Winning the buy-in of end users.

Since a lot of other end users besides the "Proof-of-Concept" site are going to be affected by this decision, you should get their basic buy-in to the process before going further. A common mistake is to wait on this step until after the Proof-of-Concept Phase when the application is proven, but by then, some of these groups may be pursuing their own alternatives.

You first need to educate this group regarding the organization's long-term big picture goals and show them how meeting these more general goals need not conflict with their local, more specialized needs.

This is where those exciting demonstrations that vendors are so good at doing become useful. You can use them to excite users and get them to buy in to the process. They should see enough that is new and exciting in the applications and the technology that they can agree that, if the Proof-of-Concept is successful, that they will be willing to make the transition.

You should make it clear that problems are expected both at the Proof-of-Concept site and in their own individual installations and that benefits may be a little bit more long term than anyone would like. A common mistake here is painting an overly optimistic picture so that when they encounter problems in the process, they lose confidence in the project. By predicting such problems and getting them to give you their support before those problems are encountered, those inevitable problems will enhance the project team's credibility instead of undermining it.

If you can only get buy-in by promising instant, painless perfection, do not go forward. You will be unable to deliver this level of satisfaction immediately. Users must understand the long-term, as opposed to immediate, benefits of this change.

6. Putting together a workable plan for the Proof-of-Concept implementation.

After you have gotten the buy-in from the end users, you need to work with your team to put together the complete implementation plan. This plan should describe each task in the Proof-of-Concept implementation, the responsibility for that task, and the timetable for the project as a whole.

Creating such a plan to avoid all of the pitfalls in moving from the mainframe is impossible without experience in similar projects. This is where the experience of your partners comes into play. They should be able to offer a template for your plan based upon similar projects in the past. It is easy to miss some of the steps in this process—such as the subtleties in setting up data communication and conversion with the existing mainframe—unless you have been through it many times before.

The plan should define the various roles in the conversion, describe resources needed on the implementation team and on the site, describe reporting and meeting methods, communication channels, how timetables will be kept updated, and how to manage the inevitable "defects" list generated during the course of the project.

7. Negotiating the purchase of needed hardware, software, and services.

This step involves the purchasing process. After you have made a plan and each outside source's role is defined, you need to know what the project, as defined is going to cost you and what the contractual responsibilities various parties are.

At this point, commitments should be made only for the next phase, the Proof-of-Concept, but some pricing issue related to later stages may also be negotiated.

Though you may buy certain software licenses from others in this process, you must identify a primary, centralized contractor to manage the project as a whole. It is best if this contractor is an outside party, independent of internal MIS and of the software vendors themselves. It is the job of this organization to secure performance from others in the project or replace their technology with alternative suppliers.

The Proof-of-Concept Phase

During this phase, the targeted application is installed at the selected site and daily transaction processing is moved from the mainframe to the client/server system. This phase involves the direct involvement of a project team, the site end users, and the solution providers. The goal is to solve the unavoidable problems discovered in implementation at a reasonable cost and in a reasonable amount of time.

Most companies look at this step as an opportunity to re-engineer the processes they are downsizing. A common failing of downsizing projects is that they incorporate the processing bottlenecks and waste that was inherent the old system. Users are comfortable with those processes and, if given complete control over the project, will probably reinstitute most of them, if the software or the customization of it, allows.

The most cost-effective process of re-engineering is really a very simple process. It does not start with an expensive, external, time-consuming professional analysis. Instead, it starts with using the application software in the simplest and most straightforward way it was designed to work. This standard way of working will not be entirely right for your company's specific needs, but it allows your users to be retrained and to try a different approach.

When they are using the software as delivered, the real re-engineering process begins as the software is adjusted to their real needs. The software should adapt fairly quickly to most of the users' real demands. In the process, users also learn the capabilities of the new software and are able to rethink their old problems in a new context. Over a fairly short period of time, they start to make use of the new client/server distributed model.

Usually, this Proof-of-Concept tailoring process takes only three to six months to get the software to a point where it is ready to be installed at other, less dedicated sites. The Multi-Site Rollout Phase that follows should take another three to six months. However, some adjustment at each additional site should be expected as well.

One critical product of this re-engineering process is the education of users to the idea that they can have greater control and effect upon how their software works. This assures that the software can continue to adapt to the organization's needs after the installation process. Usually, significant changes in the system continue to take place for up to two years as users and management test the horizons of the new technology.

Stages in a successful Proof-of-Concept implementation include:

- Site team creation, product ordering, and organization.
- Hardware and software delivery, installation, and system audit.
- System setup and administration training.
- Application setup, user security, and implementation.
- Definition of reporting to meet site and management requirements.
- Data import and conversion from the old system.
- End user training, testing, and transaction processing.
- Final scheduling of data import/exports, period closing, reporting, and maintenance.
- System cut-over during which live processing is moved to the new system.

Multi-Site Rollout Phase

Until all sites using this application are moved off of the mainframe, most of the costs associated with application maintenance remain.

During this phase, the work in application implementation from the Proof-of-Concept Phase is moved to all other sites which are running that application on the mainframe. This phase involves *the original implementation team* from the Proof-of-Concept site, users at the new sites, and local support people. When many sites are being installed at one time, the original team must be *augmented by temporary service personnel.*

The goal of this phase is to transfer the learning from the original sites to other sites with a minimum of cost and effort. Typically each installation requires less time and effort than the last as the organization climbs the learning curve and more resources are educated in supporting the new system. This phase typically lasts for from three to five months depending on the number of sites involved and the ease with which knowledge is moved from site to site. An illustration of two multi-site rollouts is shown below:

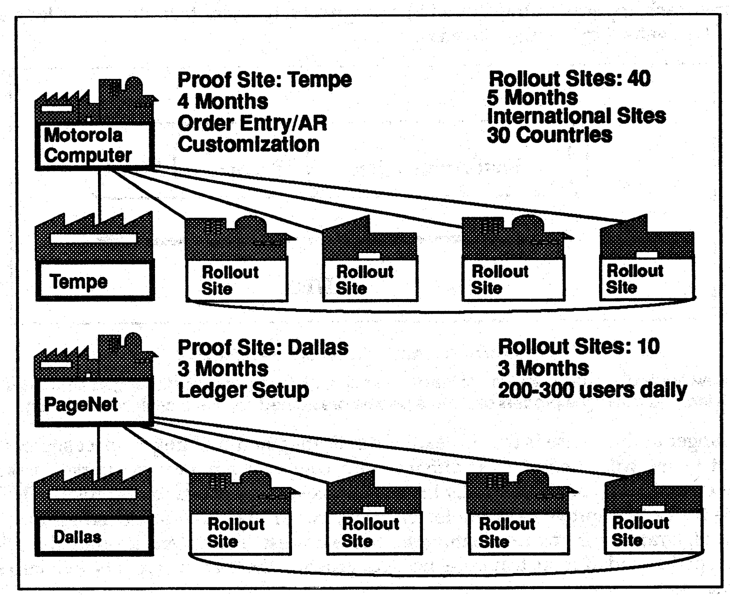

Proof Site: Tempe
4 Months
Order Entry/AR
Customization
Rollout Sites: 40
5 Months
International Sites
30 Countries
Motorola Computer
Tempe
Rollout Site
Rollout Site
Rollout Site
Rollout Site
Proof Site: Dallas
3 Months
Ledger Setup
Rollout Sites: 10
3 Months
200-300 users daily
PageNet
Dallas
Rollout Site
Rollout Site
Rollout Site
Rollout Site

Figure 2. EXAMPLES OF MULTI-SITE ROLLOUT TIME LINES

Typically, the Proof-of-Concept and Multi-Site Rollout Phases take from three to six months each for a single application. These examples are FourGen application installations.

The dangers at this phase are that knowledge will not be transferred easily between sites and that the not enough resources will be dedicated to the individual needs of each site.

The Gradual Expansion Phase

Until all mission-critical applications are moved off of the mainframe, the support of the mainframe itself cannot be discontinued.

In this phase, other applications are targeted for downsizing and the techniques learned in the first application rollout are applied to moving the other applications to the new technological base. Given that the organization now has a proven understanding of what this requires, each application transfer will be progressively easier than the last as long as the experience gained in the first move is not lost.

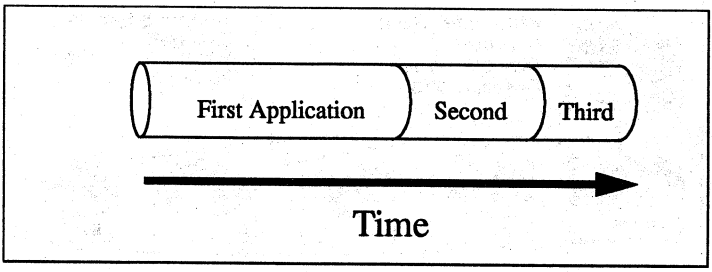


Figure 3. APPLICATION CONVERSION LEARNING

Each new application takes less time to install. The first application requires the longest time to install. As more is learned about the process and more people are trained, the installation times shorten.

The danger at this phase is that instead of duplicating and building on past success, new project teams will attempt very different and completely unproven courses of action. These experiments can dramatically increase the cost of the downsizing project. The basic rule is to try to improve each implementation, but on the basis of extending hard-won knowledge rather than by reinventing the entire process. This means keeping the original team together and, if more teams are needed, growing the team before it is split into separate teams.

This phase can take between six and twelve months depending on the number of mission-critical applications involved.

This phase also involves the spread of the application technology to new parts of the organization outside of the initial business unit. This step is critical when the mainframe has been shared by other independent business units. Even when mainframes are not shared, this step is necessary to make the project a long-term political success within the larger organization.

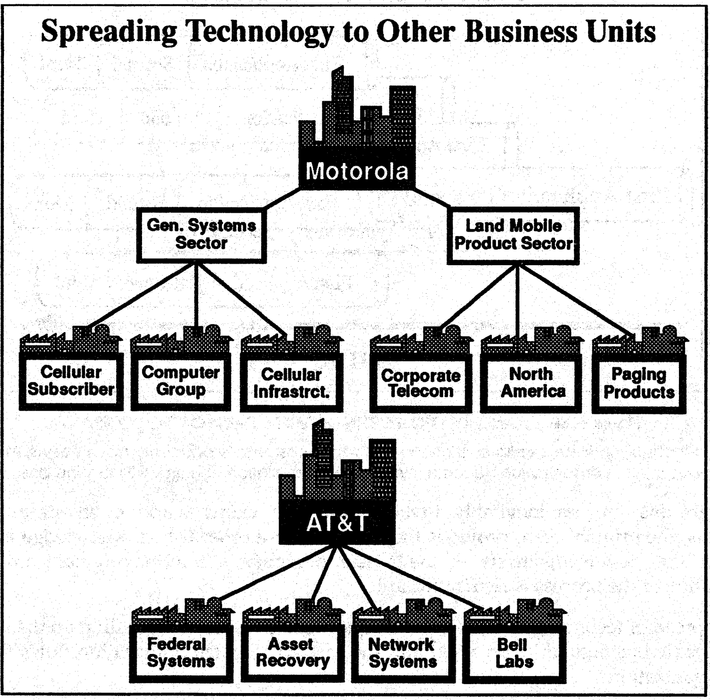

Spreading Technology to Other Business Units
Motorola
Gen. Systems Sector
Land Mobile Product Sector
Cellular Subscriber
Computer Group
Cellular Infrastrct.
Corporate Telecom
North America
Paging Products
AT&T
Federal Systems
Asset Recovery
Network Systems
Bell Labs

Figure 4. GRADUAL EXPANSION TO OTHER BUSINESS UNITS

Typically, this expansion eliminates long-term political problems and further reduces the cost of application support. The above examples are FourGen application installations.

The spread of technology within the company reduces everyone's cost of support. As knowledge about how to support the application becomes more wide-spread, the number of teams capable of installing and supporting these applications grows.

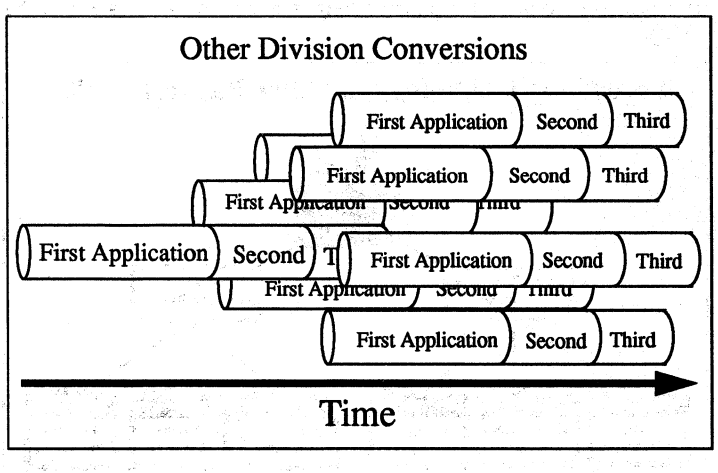

Other Division Conversions
First Application
Second
Third
First Application
Second
Third
First Application
Second
Third
First Application
Second
Third
First Application
Second
Third
First Application
Second
Third
First Application
Second
Third
Time

Figure 5. SIMULTANEOUS PROJECTS IN DIFFERENT PARTS OF THE COMPANY
When a technology is successful and offers meaningful cost and productivity advantages, more and more people in the organization become involved in its installation. This grows its support base.

Though this process inevitably involves multiple teams working on simultaneous projects, the original team provides the seed for these other teams. Knowledge is transferred to the new teams mostly by the transfer of members from the original team. Documentation of the process is also important.

This spread of technology through the company can take years depending on the number of separate business units involved and the ease with which information flows through the organization.

The Critical Elements

In meeting with large companies at the early stages of their venture into mainframe replacement, most people we see are focused on issues that have little or nothing to do with the eventual success of their project. At one such meeting with a regional airline, the MIS technicians and the operations people were embroiled in an on-going debate about the need for color monitors, mice, and fifteen-inch screens. (Surprisingly, the MIS people though color, mice, and large screens were needed; the operations people didn't want them.) No one stopped to consider for a moment whether any of these issues would spell the difference between the success or failure of the project as a whole.

Mainframe replacement projects do not stand or fall on such relatively minor issues about which workstation to use. If the organization spends twice as much as it had planned and runs two years late in the project, it won't be because it chose the wrong size monitors.

Why do such trivial issues so often dominate the debates concerning large, mission-critical project? The answer is that the people involved in the transition don't realize what the critical, make-or-break issues really are. They are nervous about these projects, so their nervousness focuses on the most visible issues instead of the most important. Usually, they don't know what the real issues are.

So, what are the critical elements in downsizing?

First, you need a good understanding the problems with your current system and a conceptual framework for understanding how those problems might be solved. Providing this understanding is the purpose of this book. The centerpiece of this understanding is a clear appreciation for the enduring importance of standards.

Next, you need a migration plan that takes into account the problems inherent in this type of project. This chapter gives a basic outline of a workable plan, but this is only an outline. In projects of this nature, the difference between success and failure is in the details, but this outline should give you an idea of whether or not you are starting in the right place and aimed in the right direction.

Next, you need the right team. This team includes people inside your company and outside experts. One of the most common mistakes made in these projects is an excessive reliance on internal resources. Most companies, no matter how large, lack the internal experience these projects require. Furthermore, internal resources aren't flexible enough to adjust to the dynamic nature of these projects as they progress. Probably most importantly, we have found that the political issues that downsizing inevitably raises are best handled by third party professionals, who are somewhat insulated from the friction of internal company politics.

Lest you think I am being self-serving in recommending outside professional to help guide these projects, let me assure you that neither I nor my company offer such services per se. We are recognized for our success in downsizing largely because we know our particular area of expertise, which is applications. We can help with this process, but we insist that our customers go elsewhere for project management. As a vendor of applications, we and all other such vendors should automatically be disqualified from playing a central role in this process.

The last critical element is the attitude of your company. The attitude that makes these projects successful is one of optimism heavily fortified by a desire to avoid risk. You must understand that the task at hand is difficult, but completely doable. You should expect and be prepared for problems and confident that you are working with people who can solve them. This attitude of trust and partnership is critical to success in ventures such as this.

PAPER NUMBER:	4021
TITLE:	The Business Case for Open Systems
PRESENTER:	Bob Lewin XOpen 1010 El Camino Real Suite 380 Menlo Park, CA 94025 415-323-7992

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Paper #4022

Title: SNMP: What can it do for me ?

Andrew J. Phillips
Technology Solutions Lab
2015 South Park Place
Atlanta GA 30339
(404) 850-2937

Abstract

The reason that SNMP was created way back when was essentially that once a network was built, it had to be managed. At first, this management process involved walking over to where a piece of equipment was physically located, and then typing commands on it's console. As networks grew more complex and spanned larger geographic areas, the process of finding and fixing problems required personnel in any location that harbored network equipment. This soon became impossible, and a technology was required that would allow management of network devices from a single location on the same network that is being managed.

One of these technologies was dubbed SNMP, the (S)imple (N)etwork (M)anagement (P)rotocol. The two key concepts being simple, so that implementation is relatively easy, and protocol, implying a standard that would allow communication and management of many varied devices. In the early years at Berkeley, even devices such as Coke machines and toasters could be "managed" using this protocol. In fact, almost ANY device can be managed by SNMP, since the process is simply to add an agent to the device that provides whatever measure of control you need for that device. This can be as simple as the toaster, or as complex as a router or a multiple CPU machine.

The purpose of this paper is to communicate both concepts and some specific practices relating to SNMP and it's use in present day environments. There is considerable documentation available on the Internet regarding SNMP, and I will reference some of the RFC's (R)equest (F)or (C)omments, where appropriate. The RFCs are a set of official documents that are available for anyone to read and form the basis for many of the technologies used in networking. See reference 2 for more details about the RFCs. A more or less complete listing is contained in Appendix A. Since this is a technical paper, the use of some three letter acronyms is unavoidable. So, I have included a glossary of terms near the end of this paper where acronyms are decoded.

I hope that the reader can gain enough insight from this material to understand how SNMP can be used, what is required in order to use it effectively, and a few examples of how Hewlett-Packard's Atlanta technology center makes use of SNMP. First, the general concepts of SNMP are covered for completeness, and then some specific ways to fit the requirements to the technology are examined, finally a few new or emerging products and technologies are mentioned. In this way, I will try to answer, in order, the following questions:

Where did SNMP come from ?
How does it work ? [overview]
What needs does it fulfill ?
What are the pieces involved ? [technical]
How can these pieces be used ? [examples]
Where is SNMP going from here ?

Background - Where did SNMP come from ?

SNMP was originally designed to help locate and correct problems on TCP/IP internetworks. It was first used to control and configure IP routers. In 1988, SNMP was chosen as the "short term" network management standard by the Internet Activities Board. This selection was due in part to the inherent simplicity of the protocol. Today, SNMP is used extensively on the Internet, and also over such diverse networks as Appletalk, Netware, DECnet and OSI. Hewlett-Packard, like many other companies, maintains an IP internet for internal use and can therefore make immediate amd practical use of SNMP technology. A good general reference is "The

SIMPLE Book", by Marshall T. Rose.

SNMP Overview - How does it work ?

SNMP typically consists of a manager that requests information from one or more "agents". The agents reside (and execute) on the devices to be managed. The manager can initiate only get, get-next, and set requests. This simplifies the implementation of agents and helps to keep the SNMP processing overhead minimal (i.e. - the "simple" part). For troubleshooting, this process can be compared to a doctor/patient interaction where the doctor asks the patient questions and the patient answers each question.

Example SNMP Interaction

```
 ----------                                      --------
| SNMP     |   ---- request ---------->        | SNMP   |
| Manager  |   <----------- reply -----        | Agent  |
 ----------                                      --------
```

The process of requesting and receiving information requires both a transport mechanism and a description of the data that is of interest. Conceptually then, SNMP can be divided into four interdependant parts. Most everyone else says three, but I think that four is both easier to understand, and more descriptive.

First is the agent, which is the piece of code, or "processing entity", that executes on each device that supports SNMP. The agents' job is basically to collect and maintain information about the device, and provide that information upon request.

The second piece is the management station, or manager, that provides the user interface to the agents being managed. Whenever I use the term manager, assume that I mean an SNMP manager, rather than a person employed in a supervisory role.

Third is the protocol itself, that is, the communication between the agents and the manager. SNMP defines both the syntax and meaning of the messages exchanged between agents and management stations. In addition to the get and set operations, SNMP also provides for a "trap", which can be thought of as way for the agent to notify the manager about some event. The trap is the only time that an agent initates a communication. One last point worthy of note is regarding security, SNMP provides a "community-name" that functions as a password. The community name is included with each manager request and is verified by the agent.

Last, we have the (M)anagement (I)nformation (B)ase or MIB, this is a hierarchical data structure that both the agent and the manager know about. The MIB structure can be compared to a UNIX or MS-DOS directory "tree", wherein one traverses one or more "branches" in order to reach the "fruit" at the end. I will use the terms object and variable interchangeably to describe the individual components of a MIB. The concept of the MIB is important because all communication between managers and agents refer to one or more MIB objects.

Example SNMP/XL Agent MIB values for a lab system in Atlanta. These variables are in the system group under the mib-2 branch, and should be available on any SNMP agent that implements mib-2.

sysDescr	HP3000 SERIES 927LX, MPE XL version C.45.00 NS Transport version B.05.05
sysObjectID	.iso.org.dod.internet.private.enterprises.hp.nm.system.4.2
sysUpTime	1:19:19:00
sysContact	John Vandegrift (123) 456-7890
sysName	R3125XL1.NAFO.HPCOM - Deep Thought
sysLocation	Technology Solutions Lab, 2nd Floor, Bldg. 2015, Atlanta
sysServices	72

Formally, the SNMP MIB is constructed so that just about any set of variables can be used for any purpose. Several organizations have input into the design, and this fact is reflected in the structure, which may seem somewhat cumbersome. One important point about MIBs is that while the RFCs define a "standard" MIB structure, there is also a "private" branch that vendors use for "extending" the MIB to include their proprietary extensions. The technical section discusses each of these in slightly more detail.

Summary - What needs does SNMP fulfill ?

The benefits derived from implementing and using the SNMP protocol can be summarized as follows. The manager makes a request of the agent, like "tell me how many errors you have seen", then the agent, after reading the proper MIB variable, replies to the query with the requested information. In this way, a dialog between a manager and an agent takes place that provides the basis for network problem resolution.

The communication protocol for SNMP, while simple, can be more or less powerful depending upon the degree of access and control the agent has over it's particular device. The management station should be flexible enough to support the MIBs, agents, and management tasks that are desired, and powerful enough to be used in real time. As shown by the Jetdirect example later in this text, an SNMP manager can be specific to one type of device, and yet still be useful.

First and foremost, SNMP was created to help in managing networks, and therefore should be judged against how well it does the job. "The job" might well be defined by how well a system can aid the individual responsible for management of the network.

I believe it makes some sense to classify this "aid" into these five management catagories:

Fault Mgmt - be able to diagnose and repair problems when they occur. This is the essence of network troubleshooting.

Config. Mgmt - be able to determine and modify a devices configuration parameters when needed.

Performance Mgmt - be able to determine what reasonable network performance is, and when there is a problem.

Security Mgmt - be able to determine if a machine is secure, and if untoward events are occuring.

Accounting - maintain a database of the nodes on the network, along with specific information about each node.

In order to achieve these aims, it is usually necessary to be able to provide historical statistics for trend analysis and establishing a baseline. Usually, the ability to support multi-vendor and multi-protocol environments is important as well. It is easy to see that SNMP is a good fit for configuration management, but the other catagories I have listed really become the function of the SNMP manager, rather than the protocol itself. Therefore, it is of paramount importance to ensure that the manager you choose fits your needs well, and can provide the features that you require.

The most visible problem with using SNMP is a lack of security. More specifically the fact that there is presently no way to "hide" the SNMP dialog between a manger and agent from anyone else on the network. This results in there being no effective way to keep unknown parties from performing get and set operations on your agents, since community names and MIB variables are visible to anyone on the local network with enough knowledge of networking and SNMP. Of course, one can disable the set function to avoid this problem , but this results in restricting the SNMP functionality to being a monitor-only process, with no way to make any changes to the devices under management. All is not lost, however, because new SNMP products and standards like SNMPv2, a remote monitoring MIB (RMON MIB), and many operating systems that support SNMP agents (Windows NT, MPE 4.0) , will address this and other network management issues. See futures section near the end for some more information about new products and standards.

Using the five point yardstick described earlier in this section, SNMP rates overall quite well. For fault management, I would say SNMP rates good to excellent. This is because many of the pieces of information necessary to diagnosis and repair a network are readily available from SNMP agents. For configuration management, SNMP again scores well. This is due to the fact that, given the proper initial setup, SNMP can not only show the network manager what the current network configurations are, but can also make real-time changes to those configurations. For performance management, scoring is difficult because this function relies more on the functionality of the SNMP manager than on the agent. Of course, given a manager that includes performance data gathering and history, this catagory would be rated strong also. Be sure to read the future products section for additional information on this subject. For security management, I would say that this is the SNMP's weakest catagory. This is because what little security features are built in only detect SNMP-related security issues, and do not provide any information about other protocols. Likewise, the protocol itself isn't designed for secure environments. Again, the section on new products provides more timely information about this issue. For accounting, this again boils down to a question of the features provided by the management station, rather than the agents. Since each agent should provide some textual information about the device, who owns it, and where it is located, I will give this an "OK" rating.

What are the pieces involved ?

The SNMP Protocol

SNMP uses UDP datagrams for communication, specifically port 161 for managers sending requests to agents, and port 162 for agents sending traps to managers. The following shows the protocol stack used in the NAFO network environment.

Session	4	SNMP :	get,get-next,set or trap ASN.1 syntax for MIB data
Connection	3	UDP :	connectionless - send a packet and hope for an answer
Network	2	IP :	the standard,routable network protocol
Hardware	1	Ethernet or 802.3	

SNMP defines both the syntax and meaning of the messages that managers and agents exchange. It uses ASN.1 (A)bstract (S)yntax (N)otation to specify both the format of messages and MIB variable names (also known as object identifiers in ASN.1). This is why SNMP messages do not have a fixed format and therefore cannot be decoded using fixed structures. The basic set of requests from the management station and responses by the agent are defined in RFC 1157. In essence, the manager functions are limited to "get", "get-next",and "set" operations, corresponding to read a value, read the next value and write a value to MIB variables. Of course, there is also the trap operation, but that is a special case and is initiated by the agent. The trap typically occurs due to some external event, like a large number of errors, and the destination to which traps are sent must be defined in advance. The manager also defines a "threshold" for traps, and a trap may be sent to more that one manager.

SNMP uses a "community-name", or password, to provide access to agents. Most agents are shipped with this value set to "public" for read access, and no value set for write access, which results in write access being disabled. There is also a trap community name and a the trap destination list mentioned above that is known by the agent. This provides the extent of security in the SNMP world today. The initialzation process for the agent provides a means of setting up community names.

The SNMP MIBs - names and numbers

The management information base, or MIB, is a term used to define the set of variables that SNMP agents and managers understand. All of the interaction between an SNMP agent and manager is done using MIB variables. For example, in the Overview section I listed the standard MIB objects in the System group along with their values for one of our machines. In the real world, any SNMP agent should support these MIB variables.

When using IP networks, all MIB references begin with the iso.org.dod.internet branch. That is, we are interested in the subtree rooted at this point. I will discuss only objects that are in this subtree, as I have little experience with the rest of the MIB definition. There are four branches stemming from this point, and they are:

dir : directory services, mostly unused today
mgmt : most used, this defines the "standard" objects
experimental : used by the internet engineering task force for testing purposes
private : vendor specific extensions under here

There are two separate MIBs in use today, MIB-1 (RFC 1212) and MIB-2 (RFC 1213). MIB-1 defines about 114 objects, and MIB-2 defines 171 objects. Objects are defined by a name and a number, and each object has access rights associated with it. In addition, many vendors add their own objects onto the private branch, increasing the number of objects even further. In general, every vendor supporting mib-2 should provide MIB objects for all of the following groups as specified in RFC 1213 :

system - This branch contains a description, the system uptime, contact, location, name and a code for the services that are available from the agent.

interfaces - This branch contains values and counters for each interface on the device (usually only one). These include status, speed, low-level address, and counters for errors, broadcasts unicasts and total traffic.

at - This is the address translation table, and consists of the contents of the ARP cache.

ip - This branch contains the IP routing table and counters relating to IP datagrams and addresses.

The other branches (icmp, tcp, udp and snmp) all contain counters and other values relating to the given protocol. To add value and enhance agent functionality, many companies add vendor-specific "extensions" onto the private branch of mib-2.

These extensions provide more control by and better features for agents. For example, the extensions to MIB-2 implemented by Hewlett-Packard include counters for each interface at the MAC, or IEEE 802.3 protocol layer, a Volume (or file system) group that deals specifically with disc drives. A Processor group that allows for multiple processors and also provides information about process tables, and a Cluster group specifically for HP-UX clusters containing information about the server and cnodes.

Since the numbers can get lengthy, and ASN.1 syntax is not a trivial concept, this notation can be somewhat confusing. To aid in clarifying, here is a short example from the SNMP/XL User's Guide (reference 7):

.... suppose one wanted to identify an instance of the variable sysDescr in MIB-2. The object class for sysDescr would look like:

iso	org	dod	internet	mgmt	mib-2	system	sysDescr
1	3	6	1	2	1	1	1

Since each field in the object class corresponds to a number, the object ID for sysDescr would be .1.3.6.1.2.1.1.1 to which is appended an instance sub-identifier of 0. Therefore, 1.3.6.1.2.1.1.1.0 identifies the one and only instance of sysDescr. The actual value for sysDescr is a string of ASCII characters, or text. SNMP deals with the exchanging of text strings in two ways. First and easiest, if the string is less than 128 bytes, its' length is given in the first byte, and the string follows immediately. In this case, the most significant bit if the length is always 0. If it was a one, that indicates that the second method is used. The second method involves coding the number of bytes that represent the length of the string plus 128 into the first byte, following that byte with the given number of bytes that are the length of the string, followed by the string itself. This results in the actual size limit of an SNMP string being imposed by the IP protocol being used, rather than any real limit being imposed by SNMP. The limit generally imposed by IP is 64 Kbytes, so an SNMP string can be up to 65535 bytes long.

For a more global view, I have included a subset of the mib-2 object "tree". This is useful for depicting the structure of MIBs. The numbers in parenthesis beside each object name show the mib number of that object.

A depiction of the structure of SNMP MIB-2

```
                         iso  (1)
                         org  (3)
                         dod  (6)
                      internet  (1)
                            |
                            |
    ----------------------------------------------
   |              |          |                   |
directory (1)  mgmt (2)  experimental (3)   private (4)
                  |
               mib-2 (1)
```

```
                    |
       -----------------------------------  ...----
      |          |               |       |          |
system (1)  interfaces (2)  at (3)  ip (4) ...snmp (11)
      |
    --------------------------- ...  -----
    |                 |                     |
sysDescr (1)    sysObjectID (2) ... sysServices (7)
```

What this really means is that SNMP uses numbers to represent MIB variable names, and vice versa. That is one reason why SNMP packets can be relatively large and only request a small amount of data.

The SNMP Agent

The agent provides all access to the devices hardware and software. Communication is in the form of requests from the management station, either a "get" - meaning reply with some information, or a "set" - meaning write some information. All this information is contained in a hierarchical data structure called a MIB that the agent manipulates (see RFC 1212 and RFC 1213).

So, the agent can be described as an entity that reads incoming SNMP requests, translates these requests into an internal (and more understandable) representation, sets or gets the given MIB value, and then sends the information back to the requestor. An agent can reply with one of the following : the objects value, a null value (0), or a noSuchName error if the requested variable is not contained in the agents MIB. The agent is asynchronous, in that the agent assumes no connection or relationship between any two requests.

In short, an SNMP agent performs the following steps for each request that it receives.

```
 Step (0) - Request arrives
 Step (1) - verify  the community name and IP addr.,
            if not OK, discard the request.
 Step (2) - convert each MIB object ID from ASN.1 into
            machine readable format.
 Step (3) - Perform the given request and save result.
 Step (4) - Convert the result back into ASN.1 format.
 Step (5) - Send reply back to requestor.
```

Of course, there is also the infamous "trap", which the agent initiates. This occures when a pre-defined threshold is reached, like 100 or more errors on an interface. At that point, the agent sends a message to the devices that it has configured as trap destinations informing them of the fact that a trap threshold was reached or exceeded, what threshold it was, and what the specific value was.

Example SNMP/XL Agents

The SNMP/XL agent provides managers with the "standard" set of mib-2 variables and functions. That is, the get,set,get-next and traps are provided, along with the object tree defined by mib-2. In order to add additional functionality, several extensions to the standard mib-2 are provided under the private branch. This path used by Hewlett-Packard is " ... private.enterprises.hp.nm". Below this, there is a system group, an interface group, an icmp and an snmp group. We are just starting to use these additional variables, and we have found that these extensions do provide significant additional information and functionality to SNMP managers. For example, there are different variables for hp-ux and MPE systems. For MPE, there is a volume group that provides a variety of information about each disc volume including percent used, largest free area, total free space, spool file usage and temp file usage. There is also a processor group specifically designed to help manage multiple CPU machines. For hp-ux, there is a fileSystem group and cluster group that can provide similar information to SNMP management stations.

To use the SNMP/XL agent, just use the SNMPCONTROL UDC to start up the agent by typing :SNMPCONTROL START. The SNMP stack on the HP3000 is normally started and stopped by the LAN transport.

Another example of an SNMP agent is the NPI agent that comes with HP JetDirect cards. This agent is "smaller" in the sense that the device it controls is simpler and therefore supports a MIB with less complexity. But, the NPI has specific needs that are particular to output devices and so supports it own set of MIB extensions (see the JetDirect example under SNMP managers).

The SNMP Manager

The SNMP manager is what the user of SNMP interacts with, that is managers normally provide the GUI as well as the functionality that makes SNMP useful in the real world. Commercially, there are several to choose from, OpenView and SunNet manager are the first two that come to mind. I have no experience whatsoever with the Sun product, and so I cannot provide any comparisons. I have used OpenView, and therefore will mostly limit my discussion to that. However, it is worth mentioning that both agents and managers are also available for PCs at reasonable costs, and more products and features are being announced all the time. In fact, the cost of adding an SNMP agent to a device is approximately $50.00, and can typically be implemented on a single chip. With this in mind, it makes little sense to purchase any network device that does not support SNMP, since the extra cost can be more than offset during the first troubleshooting experience. Furthermore, SNMP managers for PCs are quickly becoming available, with prices as low as $500.00 per copy.

For those of you with hp-ux machines, you already have access to a very useful tool called snmpwalk. This program provides the basic functionality inherent in an SNMP manager, in that it can perform get and set requests to agents. To provide user-friendly features, all one need do is create a script that calls snmpwalk to perform the required functions. For example, to retrieve the system description from any standard SNMP agent, the command "snmpwalk <device> system.syDescr " where <device> is the IP address or name of the SNMP agent's device will return the proper MIB value.

OpenView Network Node Manager Example

Hewlett-Packard OpenView provides a set of tools for network managenent that are flexible, scalable, and conform to network standards. OpenView enhances its' flexibility by providing a MIB compiler. This allows the user to fully utilize vendor-specific extensions by "compiling" the ASN.1 definition of the MIB structure into the OpenView manager. Another valuable feature of OpenView is the application builder. This provides an easy way to pre-define network management tasks and the information needed to perform them so that these tasks can be automated by the SNMP manager.

The network environment used by the NAFO sales offices is evolving from a large bridged structure towards an IP based, routed network with each office using a different subnet and all wide-area communication links terminalting in Atlanta. This network is used by over 100 HP3000s, over two thousand HP-UX workstations and servers, and a like number of PCs. There are also about 1400 terminals and hundreds of DTCs, along with the routers, hubs, bridges and printers that comprise the infrastructure. All of these computers and devices need to interoperate of course, and most network management functions, including administration, are performed remotely.

To keep a handle on this ever-changing internetwork, the network management group in Atlanta uses interconnect manager (a module for OpenView) to manage the hubs, routers and bridges that are part of HPs internal network. Their use has so far been mostly "reactive" or troubleshooting, but the features that they use are significant. Specifically, both IP and MAC layer loopback tests are useful, along with both ping and the remote ping feature to determine network conectivity. They also use the locate function to quickly "find" a node in what has grown to be a rather large network map. Interconnect manager also provides the capability to examine counters and address tables in HP hubs and bridges, which has solved some tricky problems in the past. This group also uses LanProbe II at some of the larger sites to more closely monitor traffic and develop "baseline" traffic statistics for those sites.

For a less grandiose example, I also keep track of the lab's departmental resources using the Hewlett-Packard OpenView Network Node Manager product configured to manage just our 9 or so servers and half a dozen PCs. I have found several features to be very useful, and work is still ongoing with regard to applications and integration with existing network management tools.

One of the features that I like is the fact that one can "learn" all the addresses associated with a device simply by asking it's agent. This includes MAC layer addresses, network masks (both of which are important in our environment), and what interface type happens to be configured. I also use the interface traffic variable along with the graphics capabilities of

OpenView to establish "baseline" traffic values which I can later compare and contrast. This interface traffic variable is one of the "standard" ones, and I haven't found a device that doesn't support it yet. The MIB Browser is another feature that I find particularly interesting, since one can essentially determine the structure of any vendor's mib, simply by asking. It is also useful for me to instantly get disc and CPU usage information from any server in my domain, without even having to log in. While this is true for both the HP3000 and HP9000s, it would be even more useful to be able to get this information from the PCs. Unfortunately, it appears that only Lan Manager 2.1 servers can provide this functionality, and then only by "proxy". Windows NT, however, has an SNMP agent built in that provides a significant amount of inormation about the PC.

One last item is that while it is great to be able to get information on demand, someone has to key in the values for the location and contact variables before they are useful ! I recommend that this step be included in the normal policies and procedures for adding a node to your network.

A Homegrown JetDirect Example

In Atlanta, we make extensive use of the (N)etwork (P)eripheral (I)nterface, or NPI ... this is the term given to the JetDirect interface card. We use these cards with TCP/IP protocols and use BOOTP to configure these printers from machines that are dubbed "print hubs". These print hubs control many printers, and there are several print hubs for each geographic region. This results in a large number of printers that need to be managed, and this management must be driven from the Atlanta site.

In order to manage this distributed printing in a more timely manner, a tool was developed to help the output services group. This tool is essentially a mini SNMP manager that can communicate with jetdirect cards anywhere on the network. We use only those programs that either come with the JetDirect card (i.e. hpnpadmin, hpnpf, and hpnpstat) or come with HP-UX (ping, linkloop, lp, nslookup and some others). In this way, we can provide the benefits of SNMP based device management to those who need it, without having to make any other changes to the network, or any other network management platforms that may be operating.

This tool is essentially a UNIX shell script that calls one or more of the programs listed above with the proper parameters. By using this framework, we can test functionality from the lowest protocol layer (using linkloop to the MAC address), up thru sending a testfile with hpnpf and/or using the lp spooler. This relatively simple shell script can even be used to tell if the remote printer in question is having serious problems, is improperly configured, or is just out of paper.

Since the two jetdirect programs, hpnpadmin and hpnpstat, are specifically written for JetDirect cards, they provide access to the MIB extensions provided by the NPI. In addition to displaying network statistics, counters, and textual information, once a set community name has been configured, one can even reset the interface card using SNMP. There is even a "debug" mode that shows the SNMP packets sent and received by the hpnpstat utility for testing purposes.

Future Directions - What about new products ?

More recently, a new iteration of SNMP, called SNMP version 2, or SNMPv2, has been discussed. This new version addresses the security issue by incorporating authentication, encryption, and more robust access control to MIB variables. Of course, how one chooses to use this additional functionality is dependant upon the particular situation. This version uses a "shared secret", known by manager and agent, to provide this security level.

Another relatively new product is the new LanProbe II that uses the RMON MIB (RFC 1271) and an 80386SX to process network traffic in real time. I see the RMON MIB as a way to define a LAN segment, in the same way that the original MIBs defined a router that connects LAN segments together. The RMON MIB contains branches for general segment statistics, traffic history, alarms, filters, and a traffic matrix that shows who was talking with whom along with much other data of interest.

I recently had an opportunity to make use of a PC on our local network that was running the beta version of Windows NT. This version included an SNMP agent that supported an extended MIB-II and used a DLL to incorporate the agent functionality. The agent was not only built-in but also quite easy to use and configure. The functionality demonstrated by this agent was very promising, as it provided a variety of information about the PC, including details about server usage, who is using the PC, and signal when disc free space get low or when multiple errors are detected. I see this PC agent as a way to finally include PCs as "managed" machines, like HP9000s or HP3000s.

Lastly, I will note that there are also RFCs defining standards for agents that manage RS-232 and parallel ports that can be accessed via the network. Four significant vendors, including Hewlett-Packard, have agreed to support these standards on their devices.

References

[1] "TCP/IP Ethernet Network Peripheral Interface for HP-UX and SunOS Systems Administrators Guide", HP part number C2850-90001, First Edition, December 1991.

[2] "Internetworking with TCP/IP", vol. 2, by Douglas E. Comer and David L. Stevens, Prentice Hall Inc., Englewood Cliffs, NJ, 1991.

[3] "HP OpenView Network Node Manager 3.0 Administrator's Reference", HP part number J2316-90002, June 1992.

[4] "HP OpenView Network Node Manager 3.0 User's Guide", HP part number J2316-90001, June 1992.

[5] "NT Beta Release Notes", Microsoft Co., October 1992.

[6] "Emulex Performance Series TCP/IP-LAT Protocol and Command Reference", Emulex part number ER2050006-00, Revision A, April 1992.

[7] "HP SNMP/XL User's Guide", HP part number 36922-61029, Edition 2, June 1992.

Appendix A

A Short List (in no particular order) of Some RFCs relating to SNMP

RFC 1109 - Strategy
RFC 1212 - Objects
RFC 1213 - MIB-2 Specifications
RFC 1352 - Security
RFC 1351 - The Administration Model
RFC 1303 - Convention for describing Agents
RFC 1270 - Communication services
RFC 1157 - The SNMP Standard
RFC 1156 - MIB Specifications
RFC 1155 - SMI-Structure of Management Information
RFC 1158 - More MIB-2 Stuff
RFC 1271 - RMON MIB
RFC 1318 - Parallel Interface
RFC 1317 - RS-232 Interface

GLOSSARY

Internet - a world-wide IP network begun by Dod.
IP - internet protocol
TCP - transmission control protocol, a session oriented method of communicating over IP
UDP - user datagram protocol
MAC - media access control, the lowest protocol layer
GUI - graphical user interface
linkloop - a method of sending and receiving packets at the MAC layer
NPI - network peripheral interface, the term used name HP jetdirect cards
ARP cache- a table of IP address to MAC address mappings

PAPER NUMBER: 4023

TITLE: Commercial Client/Server:
How to Develop Business Solutions

PRESENTER: Ron Rolland
Anderson Consulting

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Paper #:	4024
Title:	Getting Ready for the New Standards in Distributed Systems Management
Presenter:	Mike O'Rourke, Director of Business Programs Tivoli Systems, Inc. 6034 West Courtyard Drive, Suite 210 Austin, TX 78730 (512) 794-9070

Abstract

A number of evolving standards -- including the Open Software Foundation's Distributed Management Environment and UNIX Systems Laboratories' Distributed Manager -- will soon make it easier, theoretically, to manage distributed, heterogeneous computers.

Central to these standards is an object-oriented framework and related technologies from Tivoli Systems, Inc. Tivoli's object-oriented technology was chosen for the systems-management framework for these standards because it meets several critical requirements for distributed systems management: 1) the ability to hide the complexity of underlying networks and systems behind a single, unifying paradigm; 2) enhanced control combined with flexibility; 3) approachability by systems and network managers, as well as by less technical systems staff and end-users; and 4) easy extensibility and customizability.

This presentation will describe the object-oriented systems management framework, its role in the DME and in other emerging standards for distributed systems management, and how it will benefit systems administrators charged with managing downsized, distributed computing environments.

Discussion points include:

an update on the DME, Distributed Manager, and other standards efforts in distributed systems management

what systems administrators should do and when to plan for the DME and other standards

migration issues: how to transition your current expertise and programs to the DME and DM

organizational issues: how to sell your management on standards, how standards will change, for the better, how you use your systems management staff

what organizations can look for in the future in standards-based DSM technologies

In addition to supplying the core systems-management technology of the DME and Distributed Manager, Tivoli offers the Tivoli Management Environment, a commercially available suite of software for managing the distributed computer based on these standards. Because TME is the first systems management software compatible with the base framework of the DME, a number of organizations are using TME today to implement DME-compliant distributed systems management solutions.

• Handouts of the presentation will be available at the session. •

PAPER NUMBER: 4025

TITLE: The Open Systems Decision

PRESENTER:
Jesse Bornfreund
UNIX International
20 Waterview Blvd.
Parisppany, NJ 07054
201-263-8400

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Session #4026

Tom Axbey

Applix, Inc.

Adaptive Applications for a Client/Server Environment

Increasingly, user organizations have adopted architectural standards to ensure that new applications both address an immediate operational need, and can conform to the corporate information architecture as it evolves toward open systems. At the same time, users have grown to expect that the personal computing environment can be molded to the business practices of their workgroup or department, and that both personal and business applications should operate from one user interface. The general success of client/server computing will depend on achieving an acceptable balance across personal, workgroup, and corporate computing styles and needs. The emergence of a new class of applications - adaptive client/server applications - holds significant promise for breaking through the client/server gridlock.

Adaptive applications provide the corporate application developer with the best of two worlds by offering the immediate usefulness of packaged applications along with the tools to modify and extend the packaged applications as needed to suit the users. Adaptive applications have emerged as a new class of applications that bridge the yawning gap between the class of rigidly preconfigured desktop applications and the class of blank slate general-purpose programmers workbenches. Adaptive applications provide a basic set of applications or functions integrated with a set of tools that can be used to customize the appearance, function, interoperation, of the applications, or even integrate additional or replacement applications to meet unique personal, workgroup, or enterprise requirements.

In this seminar Tom Axbey will discuss examples of adaptive applications in the enterprise.

Future Directions for OSF User Environment Technology

Vania Joloboff
Technical Director, User Environment

Open Software Foundation, Inc.
11 Cambriddge Center
Cambridge, MA 02142
617-621-8700

Summary: OSF/Motif® is one of the few user environments available across virtually all major computing platforms. Motif is based on the X Window System technology from the MIT X Consortium, and is the de facto standard user interface for Open Systems. The presentation will elaborate on the future direction and enhancements to the OSF User Environment. It will also suggest how the next generation of user environment technology from OSF would benefit end users and software developers.

OSF/Motif® and the X Window System -- An open systems success story
    Availability across platforms
    Adherence to industry standards
    Consistency with Windows and CUA

OSF/Motif Program Status:
    Motif 1.2
    The next release... Motif 2.0, the "Motif Extensibility Release"
    Position within COSE Common Desktop Environment
    Motif specification to X/Open
    Motif progress in IEEE P1295.1
    Automated test technology and process

Challenges for Motif:
    Consistency through validation and certification
    A collaborative approach to enhancements
    Integration with other Windowing environments
    The evolution of standards

Choices for the Future:
- Structured Graphics
- Networked application interoperability
- Multimedia...

Toward the next generation:
- Human Factors innovations
- "Write-once" user interfaces
- Distributed display services
- Investment protection in current user interfaces, applications, and equipment

Copies of the presentation will be available by request.

Paper # 4027

National Language Support: Providing Data Processing Applications for the Global Marketplace

Paper No. 4028

Natasha Flaherty
Technical Staff

Oracle Corporation
HP Products Division

500 Oracle Parkway
Box 659408
Redwood Shores, CA 94065
(415) 506-7000

Abstract

Multi-national companies are becoming more prevalent in today's global economy than ever before. It is becoming increasingly common for one corporation to own several subsidiaries, or for several companies to work together in an international partnership. In these situations, language can be a barrier when people communicate with each other and computers.

The global marketplace presents some unique challenges for data processing applications. To be effective, an application must interact with the user in his or her own language. Data must be displayed in accordance with the local customs of that region. And for maximum efficiency, data must be shared enterprise-wide, among the multi-national organizations.

From a software development perspective, the task of providing multi-national software must be carefully considered. How can one build multi-national software without a multi-national staff? What is the best way to package and maintain this code? And what are some options for architecting a product that will be usable by all of the customer base?

This paper will define and explore the basic issues surrounding such National Language Support. An approach to the problem will be considered using business applications in the Financial, Manufacturing, and Human Resources arenas as an example. This paper is written from a developer's perspective but may be enjoyed by anyone wishing to learn more about National Language Support technology.

In today's global economy we see an increasing number of multi-national corporations and partnerships. Very often developing emerging markets entails going beyond national boundaries. Political events of recent years indicate that this trend is likely to continue. How will data processing applications keep up with this trend and meet the needs of the global marketplace?

First, some background about the market issues that must be addressed in order to enable National Language Support in end user applications is in order. Perhaps the most basic requirement is that applications must communicate information in the user's native language. For some multi-lingual countries, this challenge is met everyday at home with the use of multiple signs or duplicate information on product packaging. But in the case of computer applications, it would not be very efficient to place an English application, for example, in front of a non-English speaking user and expect the user to refer to a printed manual which contained translations of the boilerplate text that was displayed on the screen. Unfortunately, in most cases duplicating the information on-screen would prove both too costly in screen real estate and too distracting to the user.

Effective applications must adhere to the local customs and conventions of their users. A European colleague of mine learned the hard way about date formatting in the United States. To maintain continuous employment, he wanted the renewal of his work visa to be effective April 5, 1991. However, on the visa paperwork, he wrote the start date as 5-4-91. Despite attempts by his lawyers to rectify the mistake, government authorities insisted that he leave the country when his current visa expired, and return to the U.S. no earlier than May 4, 1991.

Currency symbols, radix symbols, and group separators are important notational conventions for financial applications. For example, the sum of one million U.S. dollars is written as $1,000,000.00, while one million French Francs is denoted as 1 000 000,00 FF. Note the different uses for the comma symbol here: in the U.S. example it is the thousands separator, while in French it is used as the radix symbol or decimal point. Additionally, we see that the French group separator symbol isn't really a symbol at all, rather, it is a space. Also, we see that the location of the currency symbol is different: in the U.S. it precedes the amount, while in France it follows the currency amount.

As another example, when using the international currency symbol, ¥1,000,000 denotes one million Japanese yen. Although one will see the decimal point used in currency exchange rates, there is no currency denomination smaller than 1 yen. Because of this, fractions of yen and decimal points are not normally written in a currency amounts. To put this in perspective, the sum of ¥10 is just enough to initiate a local telephone call. Since this is a small amount, in Japan it is much more common to see it written as 10円. Again, we see that the placement and shape of the currency symbol can differ, even in the same language!

Accommodating the myriad of business rules that apply in different countries may be one of the most difficult tasks of all. In Australia, for example, depreciation of an asset is calculated at a daily rate. Employees in the United States are usually associated with a Social Security number, while employees in the United Kingdom have a National Identification number. And of course, how you account for currency exchange rates in your business transactions can greatly affect your bottom line.

A number of technical concerns must also be addressed by applications with National Language Support. The primary one is the issue of supporting different character encoding schemes, also known as character sets or code pages. A character encoding scheme maps a particular binary value to the appropriate character shape. When a terminal receives a keystroke, it must use its character encoding scheme to determine which pixels to light up on the display. There are a number of different character encoding schemes used today. Some are hardware specific, while others have been developed by standards organizations. Sharing information can pose difficulties in a heterogeneous computing environment when different character sets are in use.

It is important to understand the relationship between character set order and a correct linguistic sort. In most instances, sorting a group of words by their binary value will not produce correct results. For example, the US7 ASCII character set is organized such that upper case letters come before lower case letters. Hence a sort using comparisons of binary values would return the results in Table 1. In the EBCDIC character set, lower case letters come before upper case letters, so a binary sort of the same data would produce the results in Table 2.

Character encoding schemes that handle European languages have similar problems when producing a linguistically correct sort. Hewlett-Packard has developed the 8-bit character encoding scheme "Roman8" to handle American English, French Canadian, Danish, Dutch, English, Finnish, French, German, Icelandic, Italian, Norwegian, Portuguese, Spanish, and Swedish. This is an 8-bit scheme where the lower 128 bytes are similar to the ASCII character set and the upper 94 bytes contain the additional characters (such as å, ä, ç, é, and ß) required to support the non-English languages mentioned above. Additionally, the £ and ¢ symbols are provided in the upper characters. 34 bytes in this 8-bit scheme remain undefined. The International Standards Organization has developed a set of character encoding schemes with the name "ISO 8859-" and a suffix numeral. The ISO 8859-1 character set supports the same languages as Roman8 and is one of the character sets also supported by Hewlett-Packard for portability reasons.

With this background information, the complexity of linguistic sorts in multiple European languages can be more readily explained. For example, the set ABC, ABZ, BCD, and ÄBC (A *umlaut*) is sorted by binary value in the HP Roman8 character set. This also happens to be the correct order for a linguistic sort in Swedish. However, in German the Ä is sorted between A and B. Hence the correct result in German would be ABC, ABZ, ÄBC, BCD.

An interesting problem happens in Spanish and other languages with double characters. The Spanish "ll" is linguistically treated as a single character that is sorted after "l" and before "m". Similarly, the "ch" in Spanish is treated as a single character which is sorted after "c" and before "d". Sorting by binary value in the

Roman8 character set gives the results in Table 3. Table 4, however, shows the results of a linguistically correct Spanish sort.

With the introduction of support for Asian languages, character encoding schemes have become increasingly complex. Most European languages can be encoded with a 7 or 8 bit encoding scheme, where one character is one byte in length. But to support the Japanese, Chinese, and Korean languages, a minimum of two bytes per character is required. Additionally, standard ASCII characters must also be contained in the character set, for operating system communication and backwards compatibility. This poses two issues that must be addressed: How can multi-byte characters be distinguished from single-byte characters in a character stream, and how can characters which take two display positions, such as most multi-byte characters, be displayed alongside single-byte characters taking up a single display position on the screen? I will briefly explain multi-byte encoding schemes in the latter part of this paper.

Cursor direction can be an issue as well. Not all languages are written left-to-right, top-down as English is. Some Middle Eastern languages, such as Arabic, are written from right-to-left, although numerals are written with the left-to-right convention and European words can be expected to appear on an occasional basis. Special terminal hardware must be used in this case, but it also requires the software developer to design with these special data display and sorting issues in mind. The challenges with Arabic are compounded by the fact that character shape can change depending on the location in the word. However, this issue is usually handled at the terminal display level. Currently, languages that are traditionally written in a top-down, right-to-left format, such as Japanese and Chinese are supported using the western format of left-to-right, top-down. This has proven to be acceptable in Asian markets.

Now that we have established some of the basic issues of National Language Support, we turn to the topic of National Language Support in data processing applications. Perhaps the most difficult scenario is that of a multi-national corporation with offices in multiple locations transacting with a single database in several different languages. In this scenario, several issues need to be addressed. First, we are assuming users in different countries share the same logical database. Secondly, since these are multi-national users, they do not share a common language. Thirdly, since users are in multiple locations, it is very likely that a number of different terminals and various hardware platforms are in use throughout the corporation. In light of these issues, it becomes clear that each user session must be configured for the user's language, locale, and terminal code page. In addition, the database must be able to accept and manipulate each users' data. This is the approach used by the ORACLE Server and other ORACLE products. Since NLS functionality has been enhanced in the ORACLE7 Server, this discussion will focus on the new methods and parameters used to configure an ORACLE7 session.

With ORACLE7, all language-dependent behavior can be specified by defining the environment variable **NLS_LANG**. This variable has three components and each component specifies a certain subset of NLS functionality for the session. It is specified in the form:

NLS_LANG = *language_territory.charset.*

The **language** component specifies conventions such as the language used for ORACLE messages, day and month names, the symbols for AD, BC, AM, and PM, and the default sorting mechanism. It also specifies default values for the territory and character set arguments, so that either or both *territory* or *charset* can be omitted. The **territory** component specifies conventions such as the date format, decimal and group separator characters, day and week numbering scheme, and default ISO and local currency symbols. The **charset** component specifies the character set to be used in that session. The value of this component is an ORACLE acronym for each supported character encoding scheme. Some examples of values for **NLS_LANG** are shown in Table 5.

An ORACLE7 Server uses database parameters **NLS_LANGUAGE** and **NLS_TERRITORY** as the primary controls for NLS functionality. When an application connects to the database, if **NLS_LANG** is set in the client's environment, an ALTER SESSION statement is automatically executed. This sets the values of **NLS_LANGUAGE** and **NLS_TERRITORY** to the *language* and *territory* parameters of **NLS_LANG**, respectively. Any default values specified in the database parameter file are subsequently overridden for the duration of the session, and for all instances that the session is connected to.

Several other parameters can be specified in the database initialization file to configure the behavior of an ORACLE7 database. They are: **NLS_CURRENCY**, **NLS_DATE_FORMAT**, **NLS_DATE_LANGUAGE**, **NLS_ISO_CURRENCY**, **NLS_NUMERIC_CHARACTERS**, and **NLS_SORT**. The default value for all of these parameters is derived from either **NLS_LANGUAGE** or **NLS_TERRITORY**.

With the use of one environment variable, a user can easily customize his or her ORACLE session to suit individual and terminal hardware needs. But how will the information entered by this user be seen and utilized by another user with a different environment? The key is to create the database using a character set that is a superset of all of the client application character sets. When a user enters data at his or her terminal, it will be automatically converted and stored in the codeset used by the server. Then when another user requests the information, it will be automatically converted, if necessary, to that user's chosen character set. If the destination character set does not contain all the characters necessary to represent the original data, specified replacement characters will be used. For example, é, ê, è, and ë might all be replaced by e if the destination character set is US7 ASCII, a seven bit code that does not contain non-English characters. And while ë will be displayed as ë in WE8ROMAN8 (HP's Roman8), WE8DEC, and JA16SJIS, the characters ナタリー from the Japanese Shift JIS character set (JA16SJIS) have no counterpart in either WE8ROMAN8 or WE8DEC, and replacement characters will be substituted. Unfortunately, there is no widely accepted character encoding scheme for all the languages of the world at the present time.

The software provider is faced with a unique challenge, "How can I provide multi-national applications without a multi-national staff?" One of the key development strategies for this problem is to separate code that the user sees from code that the user does not see. Error messages, other displayed messages, and boilerplate text should all be externalized as language dependent data from the language independent portion of the code. Like all Oracle products, Oracle Applications maintains externalized message files that can be translated independently of the software development team. With this process, product translations can progress independently of each other. A utility then converts these translated text files into machine-readable, binary message files in the desired character set. Application forms which contain boilerplate text can also be easily converted to other supported languages.

Modularity is also the key for packaging and maintaining application code. It is wise to decompose the code design such that country or region specific functionality can be separated from the generic part of the software. An Oracle Applications package may actually have up to three components. It will always have the Base product, which includes C programs and libraries, and American English forms, reports, AutoInstall text files, and demonstration database. A Language Translation portion may be included, which contains translated versions of Base product forms, reports, and AutoInstall text files for a specific language and territory. And finally, should there be any modifications made by local distributors that are not incorporated into the Base product, a Localization portion will be added, which contains either modified Base product forms and reports or forms and reports used for a completely separate and territory specific application.

When architecting applications that will offer National Language Support, several key points must be kept in mind. First, one character is no longer guaranteed to be one byte in length. With the introduction of multi-byte character encoding schemes, character handling routines must operate on a language independent basis. For example, in the Japanese JIS encoding scheme, each character is two bytes in length, with the most significant bit determining whether or not the upper byte is significant. The Japanese Shift-JIS character encoding scheme follows a different principle, using special "shift-in" and "shift-out" characters around each stream of multi-byte characters. A "shift-in" signifies that multi-byte characters are about to follow, while a "shift-out" signifies the end of a multi-byte stream. Because of the various methods used by multi-byte character sets, a function may no longer walk through a character stream by merely incrementing a pointer. This could result in only half of a character being returned or a multi-byte character being mistaken for a single-byte character. When using the Shift-JIS character encoding scheme, one must be careful not to split apart the "shift-in" or "shift-out" characters when returning a text stream. Additional space must also be allocated for text arrays, to accommodate the upper bytes and/or the special shifting characters.

Secondly, character comparison functions and sorting routines must be designed to operate on a language independent basis. We've already seen how sorting by binary value alone does not always return what is linguistically correct.

Functions that make comparisons based on the linguistic collating sequence for the target language must be implemented for proper NLS functionality.

Thirdly, functions that support proper conversions between upper and lower case must be employed. With ASCII and EBCDIC character sets, adding or subtracting a constant amount from the numerical value of the character would produce the corresponding upper- or lowercase character. For example, in US7 ASCII, the value of "A" is hexadecimal 41 (41H), while the value of "a" is 61H, a difference of 20H. Hence, since "B" has a numerical value of 42H, "b" is guaranteed to have the value 62H. For ASCII, to find the lowercase of a letter, just add 20H; to find the uppercase of a letter just subtract 20H. In the EBCDIC character set, the ordering and the constant value are different. "a" has the value 81H in EBCDIC, while "A" has the value C1H, a difference of 50H. Since the ordering is reversed, to uppercase a letter in EBCIDIC one must add 50H, while to lowercase a letter simply subtract 50H.

With extended character sets, there are no fixed offsets for the non-ASCII characters. Table 6 gives numerical decimal values for eight characters in the Hewlett-Packard Roman8 character set. After doing the subtraction for each example, we find that the decimal difference between À and à is 39, between Â and â is 30, È and è gives 38 and Ê and ê is 29. Clearly, there is no linear relationship between the international characters in this character encoding scheme. Other encoding schemes yield similar results. Additionally, some languages have various exceptions to the rules for uppercasing characters. For example, in European French diacritical marks on uppercase characters are usually dropped, while in Canadian French, diacritical markings are usually retained. For these reasons a language independent uppercase/lowercase function must be employed.

Finally, character conversion routines must be developed to convert characters between different character encoding schemes. As we have seen, there is no one universal character set which encompasses all languages and is used on all hardware platforms. Until this is the case, data will have to be converted between character sets in order to be shared across platforms or countries. If a certain character is not available in a target character set, an intelligent substitution should be made. For example, e could replace é, ê, è, and ë, while u might be an acceptable substitute for ü. However, it should always be recognizable when a substitution has been made.

To effectively design software for the global marketplace, it is important to separate the language-independent and language-dependent portions of the code. Some of the basic assumptions that held true for either ASCII or EBCDIC character sets are no longer true when international character encoding schemes are considered. By functioning well in a heterogeneous computing environment, flexible and adaptable software can help break down the language barrier when people communicate with each other and computers.

Tables & Figures

Table 1

Chicago
Phoenix
bears
suns

Table 2

bears
suns
Chicago
Phoenix

Table 3

chaleco
cuna
día
llava
loro
maíz

Table 4

cuna
chaleco
día
loro
llava
maíz

Table 5

AMERICAN_AMERICA.US7ASCII
FRENCH_FRANCE.ISO8859P1
FRENCH_CANADA.WE8DEC
JAPANESE_JAPAN.JA16EUC

Table 6

À	161
à	200
Â	162
â	192
È	163
è	201
Ê	164
ê	193

Bibliography

Freeman, Roger L. *Reference Manual for Telecommunications Engineering*. John Wiley and Sons. 1985

Hewlett-Packard Company. *Native Language Support: User's Guide HP 9000 Computers*. Second Ed. 1991

Oki Electric Industry Co., Ltd. *MS-DOS 3.1 Usage Guide*. 1987

Oracle Corp. *ORACLE RDBMS Database Administrator's Guide, Version 6.0*. Appendix D, The INIT.ORA Parameters. Appendix F, National Language Support. Rev. 1990

Oracle Corp. *ORACLE7 Server Administrator's Guide*. Appendix A, Initialization Parameter Files. Appendix C, National Language Support. 1992

Oracle Corp. *ORACLE7 Server Application Developer's Guide*. Appendix E, National Language Support. 1992

Oracle Corp. *ORACLE7 Server Concepts Manual*. Appendix A, National Language Support. 1992

Oracle Corp. *Product Internationalization: Porting and Packaging*. January 7, 1993

Wharton, Kevin. "Catering for Language and Territory Conventions in Applications", Paper #232, *International Oracle User Week Proceedings*, Vol. 1. 1992

ORACLE and ORACLE7 are registered trademarks of Oracle Corporation.

Business Issues of EDI:
12 Steps to Successful EDI Implementation

by Phyllis Sokol, Director
and John Stelzer, Senior EDI Consultant
COMMERCE:Institute

EDI implementation is a twelve-step process that consists of three main phases: planning, implementation, and leveraging your company's EDI investment through expansion of your EDI activity. This article will discuss each of these three phases. This article is a condensed version of a full-day course, Business Side of EDI Implementation, we offer at the COMMERCE:Institute. Because the description of EDI implementation that we provide here is an abbreviated summary and not a fully detailed account, this article is not recommended as a complete guide to EDI implementation, but rather is intended as an overview for companies doing a preliminary investigation of EDI.

The first section focuses on the initial planning phase of EDI implementation. This phase is made of up six steps. Steps 1 through 3 describe how to build the business case for EDI. Once you've done this necessary groundwork, you're ready for Steps 4 through 6, which tell how to develop an implementation plan to gain the necessary commitment and funding from your company's upper management. Our approach to both stages is practical and task-oriented. In this article, some of our discussions are directed toward business managers, while others are aimed at technical support staff. We take this approach in our conviction that successful EDI implementation requires a partnership within your company of both business and technical interests.

Steps 1 through 3: Building the Case for EDI

Step 1: Conduct the Initial Business and Technical Analyses

The first step in building the case for EDI involves achieving a thorough understanding of the business environment in which EDI will function. This understanding is the groundwork for successful EDI implementation. During this step, it is imperative that both business and technical managers analyze the issues that pertain to them.

BUSINESS TASKS: To start on the road to successful EDI implementation, business managers need to document business activity as it exists without EDI. This facilitates the identification of areas within the business environment that may need improvement and that, therefore, provide excellent opportunities for utilizing EDI. Several key tasks are required of your company's business managers during this step:

Task 1: Evaluate the Activity of Each Functional Business Area. Each functional business area within the company will identify the external organizations with which it exchanges information, and will note the type of information and the mode of exchange. For example, the purchasing department should list suppliers as the external organization with whom they do business and purchase orders, follow-up phone calls, faxes, etc. as the business transactions. They should then identify the specific pieces of information currently provided in each of the business transactions as well as all of the manual tasks that are currently performed to support the business transactions.

Task 2: Develop a Procedural Flow for Each Area. Each functional business area should draw a procedural flow chart based on the outside organizations, transactions, business information and manual procedures gathered in Task 1.

Task 3: Identify Your EDI Opportunities. With the flow chart in hand, you are now prepared to identify instances where a quicker movement of information is needed, where more complete and accurate information is required, and where overall efficiency could be improved by allowing computer applications to make decisions and conduct editing currently done by people.

TECHNICAL TASKS: Just as functional business managers must evaluate each business area to identify potential applications of EDI, the technical staff must evaluate the existing systems and communication environment. Several key tasks are required of your company's MIS managers at this time:

Task 1: Evaluate Each Functional Area's Systems and Communication Capabilities. This task is crucial to determine what functionality exists in application programs today, and to identify inter-company communications capability (relevant because EDI uses telecommunications lines).

Task 2: Develop a Data Flow for Each Business Area. Just as each functional business area drew up a procedural flow chart, MIS also needs to draw a data flow diagram for each computer application used the functional area. This will show inputs and outputs of the application, as well as all system functionality.

Task 3: Identify EDI Hardware and Software Options. Finally, MIS managers should identify the computer and communications hardware and software currently in place, to evaluate their EDI hardware and software implementation options and to develop an estimate of time and dollars for implementing EDI.

Step 2: Recommend the Scope of the Proposed EDI Project

Once the business and technical analyses are complete, it is possible to establish the scope of the proposed EDI project. The business and technical data flows should now be combined. This consolidation will point out any difference in the understanding of the two groups, and such discrepancies can be dealt with in arriving at a common data flow diagram. With this comprehensive data flow diagram in hand, you can base your recommendations for streamlining business procedures and automating currently manual tasks on actual, documented need, rather than on speculation and guesswork.

It is important during this step to develop an EDI project scope large enough to predict real benefits, while keeping the project small enough (at least initially) to be realistically workable at the outset.

Step 3: Get Preliminary Management Approval

Using your combined business and technical data flow diagram, you can develop the business case for EDI and gain senior management's approval to continue the planning stages for EDI implementation. This preliminary report is not the EDI implementation plan--that plan will be developed later, in Step 6. Instead, in the current report, you are building a business case showing how EDI will respond to real business issues that exist today. This will be used to encourage management to make the investment in a full and detailed EDI implementation plan.

In your preliminary report to management, you should be sure to address the following issues:

Current Problems and Inefficiencies
Business Impact of These Problems and Inefficiencies
Recommended EDI Project Scope
Expected Results
Estimated Costs

Steps 4 through 6: Gaining Management Commitment and Funding for EDI Implementation

Step 4: Define Business Needs for the Final Proposal

This step is directed at business managers. In performing the four tasks outlined here, business managers will be investigating their information needs, the potential benefits of using EDI as a productivity tool, and what will be needed to make the business staff trust the EDI system.

Task 1: Identify Business Information Requirements. For this task, business managers in EDI sender organizations need to identify all pieces of information available to be sent. EDI receivers need to identify the information that is required by their application systems in order to process the business transaction being received.

Task 2: Identify Audit and Security Requirements. We believe that the needs for tracking, controlling, accessing, and maintaining information are identical for paper-based and EDI systems; however, techniques used in the paper-based environment are not applicable to EDI. During this task you will need to identify the control points and information access needs in each business area in which you are planning to implement EDI.

Task 3: Identify New Procedures and Policies That Will Accompany EDI Implementation. Using the business information and audit/control requirements identified in Tasks 1 and 2, you can identify new procedures and productivity tools (often these include on-line systems) that are needed to support the EDI environment.

Task 4: Define Business Specifications for MIS. EDI productivity tools such as on-line systems are ultimately developed by a company's technical staff. Business managers need to develop business specifications to describe the functions they expect these tools to perform. Technical people will use these specifications to insure that the systems will conform to business needs.

Step 5: Define Technical Needs for the Final Proposal

Just as the business managers defined their needs for EDI, so must the technical staff. The following three tasks help MIS managers devise a plan that will turn business requests into system realities.

Task 1: Analyze Business Specifications for System Implications. The technical group must evaluate information requirements of the proposed EDI system for: current availability of information in system files, ability to develop new files and databases, compatibility with existing applications, and availability of computing resources.

Task 2: Evaluate Implementation Options. Based on the computer resources currently available in-house in conjunction with the functionality required by the business community, the technical staff will choose the size and sophistication of the EDI translator that best fulfills the company's EDI requirements.

Task 3: Develop Technical Recommendations. The final task in this step is for the technical group to draw up their EDI implementation recommendations. These recommendations will then be incorporated into the final implementation plan to be presented to management in Step 6.

Step 6: Seek Management Approval for Implementation

This final step of the planning stage for EDI implementation is a joint venture between business and technical interests. The final proposal should include both a short-range and a long-range program to support the company's business needs and to work within the company's existing computer and communications environments. To the greatest extent possible, the goal here is to draw up a clear implementation plan that will accomplish the needed improvements that were identified and presented to management in Step 3.

This comprehensive EDI implementation proposal should contain both business and technical components. Not only should it recommend software and possibly hardware purchases, it should include a plan to enhance existing programs, to develop new ones, and to design and implement new business procedures and job tasks, as necessary.

Once your EDI proposal has gained the support of upper management in terms of financial allocations and commitment of internal resources, you are ready to move ahead into the initial implementation phase.

The second phase of successful EDI implementation is the actual installation and testing of EDI within a pre-established supportive business and technical environment. During this phase, the company follows through on the proposed EDI implementation plan and runs an EDI pilot program with two or three trading partners. This phase is made of up four steps. These four steps involve developing both a business and a technical environment to support EDI; installing and testing the EDI system; and running and evaluating the EDI pilot program.

Steps 7 through 10: Implementing EDI

Step 7: Develop the Business Environment to Support EDI

Step 7 is made up of three specific tasks to be performed by the company's business managers:

Task 1: Define Business Information Requirements. To prepare the business environment for EDI, business managers must first define all information the company will need to send and receive. Because EDI is always a joint venture between a company and its trading partners, it is necessary to examine business information within the context of this partnership.

For example, in theory the EDI receiver is the one who defines the information requirements for a given transaction set. Typically, however, the more influential or EDI-experienced trading partner is the one who defines what information is sent and how it is represented in the EDI standard data format.

In completing this task, you'll want to develop a list of the information fields that you as a receiver will need, and that you as a sender have available to send. You can present this list to each prospective EDI trading partner, and each partner will, in turn, present a similar list to you. Partners resolve differences by presenting their case and negotiating.

Task 2: Define Business Procedures and Productivity Tools. EDI will be embraced by a company's employees only when it makes it easier for people to do their jobs. In performing this task, you should consult the actual staff members who will be using EDI and EDI information. Ask them to identify their needs for performing their jobs more effectively. Clearly define and then implement the business procedures and productivity tools that are needed.

Issues to be considered will include the following:

- Without manual procedures, how will business activity be tracked?
- How will data exception problems be resolved?
- How will new EDI trading partners be brought on?
- Who will be responsible for tracking EDI data?

Task 3: Define Business Training and Support Needs. A company newly involved with EDI needs to be sensitive to the fact that its business staff may be unfamiliar and uncomfortable with the new technology and may even feel threatened by it.

The company should thus be prepared to explain to employees how EDI: a) will make them more productive, b) will eliminate some of the routine elements of their current jobs, and c) will provide the opportunity to use their skills for more creative, decision-making tasks. During the early stages, a company should be sure to explain not only what EDI will do for the organization, but what EDI has to offer the employees themselves.

Step 8: Develop the Technical Environment to Support EDI

Step 8 is made up of four specific tasks to be performed by the company's technical managers.

Task 1: Define Standard Usage. The technical staff evaluates business information needs in the context of the EDI standard. Sometimes the technical group may suggest more streamlined ways of doing business via EDI. This may mean that while information requirements of the business community will be fulfilled, actual data fields transmitted and/or received via the EDI standard may appear differently than was requested by business interests.

To fill information needs of EDI transaction sets, the technical group may need to search in-house computer files. They will also need to locate places to store incoming EDI information.

Task 2: Make EDI Translator and Communication Decisions. The company's technical staff will evaluate specific EDI translator products. Now that the computer environment, required features, standard, transaction sets, and data fields are known, the technical group can select the product that best fulfills the company's EDI needs.

Task 3: Develop and Implement Application Link Software. Based on the requirements for information, additional editing, incorporation of audits and controls, and on-line productivity tools, the technical staff can develop and implement application link software and other enhanced functionality. The technical group should be sure to check with the business staff to make sure that all of their requirements are included in the overall design.

Task 4: Define Technical Training and Support Needs. Like the business staff, the technical staff will require training in EDI. They will need to understand the new products and application programs, as well as EDI in general, the standard being used, and their individual responsibilities in support of the new system.

Step 9: Install and Test All Elements of the EDI System

The new EDI system will be comprised of a variety of individual components, including:

- Application Program
- Application Link
- Associated In-House Computer Files
- EDI Translator

- Communication Hardware and Software
- EDI Third-Party Network

It will be necessary to test each of these. In so doing, you will follow the flow of information from and through the application and translator programs, verifying that all information is being passed as intended and that business requirements are being met.

During testing, keep in mind that you are not merely trying to establish that the system works. Instead, it is recommended that you try to actually make the system break down by introducing all possible problems and errors. As you do so, check the system's ability to identify errors and route them to the appropriate person or department for correction.

As part of the overall testing, you should also examine communications between your company and your EDI third-party network. In so doing, test your communication hardware and software, as well as your ability to receive, understand, and use third-party EDI control reports.

Step 10: Run and Evaluate the Pilot Program

In setting up a pilot program, you should be sure to choose trading partners with whom you already have a positive working relationship. You should also make sure that your chosen partner is willing to devote the time required to verify that EDI is functioning properly.

Look for partners with whom you transact a good deal of business. By succeeding with such a partner, you'll experience a noticeable initial reduction of manual processing. You should also select a partner with EDI experience, but if this is not possible, choose a partner with a strong desire to do EDI successfully.

During the pilot project, you'll develop a list of steps for you and your pilot trading partner to use in testing all aspects of the EDI trading relationship, both when sending and receiving.

Finally, be sure there is a EDI coordinator assigned within your company who is responsible for the pilot program, and a responsible counterpart for this person within your pilot trading partner's organization.

Once your EDI system has been installed, tested, and piloted with one or two trading partners, you are ready to move ahead into the final phase of leveraging your EDI investment. In third phase, we will be examining how to leverage your EDI investment into maximum benefits. During this phase the two main activities are 1. to expand or roll out, to as many trading partners as possible, the program you have just implemented and 2. to look for ways to implement additional EDI applications throughout your organization.

The overall goal of this third phase is to reduce the work needed for each new trading partner and each new EDI application, so that each new partner and EDI application will be easier and faster than the previous ones. By taking advantage of the work you have done in implementing EDI thus far, you'll be on the right track to accomplishing the goal of reducing implementation work and costs.

Steps 11 through 12: Leveraging your EDI Investment

Step 11: Roll Out the EDI Pilot Program

Step 11 uses the results of the tasks we specified in Step 10 that evaluated the results of your pilot program. Verifying that EDI data is being generated, transmitted, received, and handled and processed in a timely and accurate manner is essential as you bring on each new trading partner. Once you have tailored the plan to work for you, it's time to add more trading partners. The pilot plan you have developed will help you estimate the time and effort needed to bring up each new partner. Questions you'll need to ask in your evaluation process include: How long does it take to add each new partner?

How many staff members does it take to complete the required tasks? How many people are already assigned to the EDI project team? Answering questions such as these will help you to decide how many trading partners you can bring on in a specified amount of time.

Once you've computed the number of EDI trading partners you can reasonably expect to bring on board in a specified time period with your current head count and productivity level, you may find that you are unable to attain your organization's EDI goals. To increase your rate of partner implementations, you may need to increase the human resources assigned to your EDI roll-out project or, alternatively to enlist the marketing and implementation assistance of your EDI third party network services provider.

Your third-party EDI provider can also help you in marketing EDI to reluctant trading partners. Sometimes a company discovers that, once it is ready to expand its program to include new EDI trading partners, such partners are hard to find. Some trading partners may be unwilling to implement EDI. Others may be willing, but are reluctant to expend the necessary financial and human resources required for EDI implementation. Your third party network can facilitate trading partner implementation by marketing EDI products and services directly to your trading partners on your behalf, saving you the time and costs associated with developing your own EDI marketing program or increasing the size of your EDI team..

Offering EDI education to your trading partner base is another method that has proved useful in bringing on partners that are reluctant to implement EDI. In some cases their reluctance may be based on ignorance or fear of the new technology, and a few hours of EDI education can overcome these obstacles.

Sometimes, however, the above-mentioned methods will simply not work in persuading trading partners to become EDI-capable. When this is the case, it may be necessary to increase the amount of business pressure you exert by getting your top level management to openly express your EDI commitment, or by making it clear that you intend to show preference to EDI-capable trading partners.

If you are in an industry that has already made a commitment to EDI implementation, you will be likely to find many trading partners with whom you can implement EDI in a very short period of time. When potential EDI trading partners are in abundance, you will wish to prioritize them in accordance with four key factors.

- The first factor to consider is the amount of business you transact with them. Obviously it will be to your advantage to bring on your biggest trading partners first.
- The second factor is the level of EDI experience your potential partner possesses. Starting with partners who are EDI-capable and have the most EDI experience will help you identify, minimize, and even eliminate possible errors and processing exceptions that may occur later on.
- The third factor is the willingness of the partner to implement EDI. Partners who are committed to EDI success and who are willing to devote their time and effort to EDI implementation and testing will be easier to work with and will results in your most effective EDI trading partners.
- Finally, the fourth factor is the importance of the partner to your overall program. In some cases, you may have a partner that is reluctant to become involved with EDI even though you consider them a strategic partner to your long range success in EDI. In such a case you will need to follow the procedures outlined above for dealing with unwilling potential EDI partner.

Step 12: Leveraging Your EDI Investment

Step 12 involves increasing your EDI activity. As discussed in Step 11, using your EDI investment to the greatest advantage throughout the company means expanding your program to as many trading partners as possible. There are, however, other ways to take advantage of the EDI work you have accomplished and the investment you have made thus far.

One particularly cost-effective way is to increase the amount of EDI business you transact with existing EDI trading partners. Once a partner is EDI-capable, you should plan to transact more business electronically with them within the same functional business area. Doing more EDI with current partners in the same functional area is the most cost-effective way to develop your EDI program as you re-use both business and technical expertise in both partners' organizations. In addition, by adding more electronic transactions between yourself and a trading partner, you are further streamlining business procedures between the companies, thereby decreasing further the costs of doing business for both parties of the initial EDI implementation.

Another strategy for increasing EDI activity with current EDI trading partners is to add new applications in other functional business areas. For example, if your purchasing department is already sending EDI purchase orders, you should begin to receive invoices into your accounts payable department.

If the new functional area, in this case accounts payable, does not yet have the application required to support the EDI transaction, additional software development work will be needed to make this department EDI-capable. Therefore it is essential that you return to phase 1, planning, for implementation of EDI in this new functional area You will also need to define the business needs and specifications for this department in order to incorporate the new system functionality and business user productivity tools needed. If the support group that handled the initial EDI application is still available to handle this new application, then its implementation will progress more rapidly than the first one did.

A third strategy for leveraging your EDI investment involves doing EDI within other divisions of your company. Often overlap exists between the trading partner bases of the various divisions in a given company. Once EDI has been implemented at one division, substantially less effort is required for a second division to implement EDI and bring on EDI trading partners.

When implementing EDI in other divisions within your organization, it is often not practical to share the EDI software and communications links that you have already installed. Therefore, an additional investment of people and dollars may be required. Moreover, it's usually a good idea to allow the original support organization to act as consultants to the other division, since this increases the speed and efficiency of the other's EDI implementation.

Conclusion: Achieving Maximum EDI Benefits

As you move ahead with your EDI program, your goal should be to take advantage of both the people and the hardware, software, and communications structure you've already put into place in implementing EDI.

Those companies realizing the greatest EDI benefits are those that share the following characteristics:

- They have implemented EDI with a large portion of their trading partner base.
- They are trading electronically for a large portion of the dollar value of their business.
- They have incorporated EDI into several functional business areas and divisions of their company.
- They are using EDI data constructively and creatively throughout their organization.

By following the plan we've outlined in this article--developing an implementation plan based on real business needs, implementing EDI in such a way as to realize real benefits, and leveraging your EDI investment throughout your organization--your company is sure to become one of those who are gaining a major competitive advantage through the use of EDI.

Interex - 93: 4030

Computer Associates, Int'l.

White Paper

Systems Management For UNIX

Is UNIX Ready For Commercial Applications?

Five Questions Every CIO and IS Manager Should Be Able To Answer

For two decades, UNIX was used primarily in scientific and engineering applications, or for university research. It offered little that was of interest to information systems managers in commercial organizations or government. During the last three years, however, a new commercial UNIX market has emerged. At over $8 billion in 1993 and growing at over 25 per cent per year, the commercial UNIX market will soon surpass the scientific and engineering market.

Each CIO and IS manager must decide whether and when to begin to use UNIX systems for commercial processing. This White Paper focuses directly on one of the most important issues faced by those decision makers: how can UNIX systems be as reliable and secure as the legacy systems they are to replace.

Question 1. What changed to make UNIX systems acceptable for commercial processing?

The principal change that took place was that all three barriers that had slowed the growth of commercial UNIX disappeared. The three barriers were (1) hardware limitations, (2) lack of commercial software, and (3) limitations in system management.

1. Hardware: Initially, UNIX-based hardware was usually too small and underpowered for major commercial applications. Today, however, UNIX-based computers rival IBM mainframes in transaction processing speed and may actually exceed mainframes in reliability.

2. Software: For decades, only a very small number of industrial-strength commercial applications were available for UNIX. Most successful independent software vendors built their packages for IBM mainframes and other proprietary platforms. But beginning in the early nineties, in a massive shift, the majority of all the ISVs began moving their applications and building new applications for UNIX-based open systems. Today, more and better applications are available for UNIX computers than for most proprietary systems.

3. Systems management: The final barrier to commercial UNIX was weak systems management software. Without solid, bullet-proof systems management, most CIOs were reluctant to bet their businesses on UNIX. In 1992, the Meta Group explained the depth of these feelings, saying the lack of systems management "has become a show stopper issue."

Question 2. What types of problems arise when systems management is not in place?

Without systems management, UNIX computers are vulnerable to a raft of problems that arise whenever commercial processing is implemented in a new environment. The same set of problems plagued IBM mainframe users twenty years ago when commercial processing was first implemented there. One experienced IS manager said moving to UNIX was "like going back to the Dark Ages" because he was experiencing all the same problems he had faced decades earlier.

Any of these problems can be career-threatening for IS managers when they impact a high-visibility project.

- Backup tapes can be accidentally but easily destroyed when systems do not have tape protection software that guarantees only the correct tapes are used. Without tape protection data can be irretrievably lost. When that happens to critical financial or customer files, the results are very unpleasant.

- All processing can stop when file systems fill up. Without intelligent software that senses and avoids imminent over-filling, systems can fail. When this happens to remote systems, staff must often by flown to the site to correct the problem. In the process, processing can be stopped for hours.

- Auditors can prepare very negative reports on information processing projects when ultimate security is put in the hands of super users. Cliff Stolle, the famous Berkeley scientist who discovered the German hackers breaking into U.S. Defense Department computers, likens the UNIX super user to an apartment manager who has a pass key that lets him into every apartment where he can do anything he likes and never leave any tracks. Without industrial-strength security -- far in excess of what is commonly available in UNIX, your system has someone walking around with a pass key to every piece of data.

- Data can be ruined when jobs are run in the wrong order. In commercial processing, tasks have to be run in the right order. One Chicago-based securities firm, for example, reported that its system administrator forgot to start a set of jobs early enough. When UNIX's cron scheduler program started another job that was supposed to use the results of the jobs the operator had delayed, data was ruined.

- In many UNIX environments, there is no user accountability and users cannot be charged for their computing because there was no convenient invoicing system that could match the user hierarchy;
- User problems are sometimes overlooked when there is no help-desk or problem tracking system;
- Very expensive operators are needed when systems do not have automated operations technology that lowers the administrative burden.

These are just a few of the more challenging frustrations facing managers of UNIX computers. For a more complete list of the management challenges, ask your Computer Associates account manager for a copy of the booklet "Managing UNIX Computers Effectively and Efficiently."

Question 3. Are there additional problems that occur when systems are distributed?

The problems listed in the previous answer are characteristic of commercial processing environments whether they are distributed or centralized. Today's distributed systems create new management headaches, as well. As computing power is distributed, extra people have to be hired to manage remote systems, especially when because there is no operations automation technology that could automatically respond to messages that reached each system console. One experienced UNIX manager told a 1992 UNIX Expo audience, "Our UNIX computers cost twice as much to manage as our mainframes," because the automation tools are not there.

Distributed systems create pressure for single-point console automation to reduce administrative costs, for network-wide, common security access, for multi-platform job scheduling, and for network-wide accounting and charge back. In other words; the problems faced on mainframes are tougher to solve in distributed environments.

Distributed systems are usually heterogeneous. Rarely are all parts bought from a single software vendor. Hence the systems management requirement in distributed environments demand that management tools work on all the different platforms. And that demand carries over from all the varieties of UNIX to Novell Netware, to IBM's OS/2, and eve to Windows NT.

Question 4. Why didn't the UNIX vendors solve these problems?

Most UNIX vendors concentrated on scientific computing or on very small commercial applications. In these environments, most machines have only a few users, security is not a big issue, and a single administrator can juggle the tasks and keep the system operational.

It is only in the past three years that hundreds of large companies have begun evaluating UNIX for their major business applications -- including customer service and sales applications and financial and manufacturing applications. These applications each have dozens or hundreds of users. If they fail or falter, the entire business stops.

But such concerns are new to UNIX vendors. Few of the people running the vendor organizations had been involved in commercial processing. They didn't know that systems management was necessary, so they didn't provide tools to protect against the problems.

And the vendors' traditional customers, UNIX system managers, reinforced the fallacy. Few UNIX system managers want to deal with the highly structured and controlled environments necessary for safe commercial processing. It is easy to find UNIX administrators saying that rigorous controls are "overkill." Their misconceptions are based entirely on the fact that they never ran a commercial facility, never received the 3 a.m. phone calls telling them the system had failed, never had to stand in front of a vice president or president and be told that they had caused major damage to their organizations.

System management controls are put in place not because commercial system managers are "control freaks." They are put in place because there is no way to run a reliable commercial processing facility without them.

Today the UNIX vendors have fully discovered the need for system management. And, surprisingly, they have all reached the same conclusion on how to meet the need.

Question 5. What tools are available to make UNIX systems safe for commercial processing?

The ten top UNIX system vendors, along with Microsoft for Windows NT, IBM for OS/2, and Novell for Netware, have all announced support for CA-Unicenter to help make UNIX computers as well managed and bullet-proof as mainframe environments. Each is taking a different approach to making it available to clients. HP, for example, is bundling a trial copy of CA-Unicenter with all midrange and large UNIX servers shipped between July 1993 and July 1994.

The vendors are not alone in their endorsement of CA-Unicenter. UNIXWorld magazine awarded it the coveted Best Product of 1992 award and Corporate Computing magazine chose it as one of the Most Important Technologies for 1993.

CA-Unicenter solves the problems of commercial systems management fully and effectively in large measure because its developer, CA, is the same company that provides commercial systems management software for the mainframes in more than 90 per cent of the companies listed in the Fortune 500. Although CA developed CA-Unicenter by writing completely new code, every component reflects the lessons learned in more than fifteen years of helping organizations manage commercial processing environments.

Here are a few of the highlights of what CA-Unicenter can do for your UNIX environment:

1. Extends UNIX security so that your auditors will feel comfortable with UNIX for commercial processing.

   CA-Unicenter enhances login and access control and makes it all policy based. It limits access of any user, even the superuser when that is necessary. It has time-of-day controls, and security monitoring and phase-in facilities that are unique. And the policy-based design radically reduces the effort needed to manage security for large numbers of users.

2. Ensure your jobs run in the right order to avoid data corruption.

   CA-Unicenter's job scheduling component has predecessor and successor controls that allow jobs to run only when all the necessary pre-requisites have been completed. It also allows trigger-based scheduling so that tasks can be started whenever a key event occurs.

3. Protects tapes from being accidentally overwritten.

   The tape management controls ensure that UNIX doesn't write on a tape unless it is approved. It ensures that back-up tapes are kept the requisite number of days or versions, and automatically checks tapes that are mounted so that backups can run unattended.

4. Helps avoids system crashes caused by file systems filling up.

   The storage management system in CA-Unicenter automatically monitors file systems and, when file system loads reach user-determined thresholds, automatically archives files. A unique and powerful component of this system monitors all UNIX file requests. If any program requests a file that has been archived, CA-Unicenter automatically suspends the requesting program, retrieves the file, and continues the program. And all this is transparent to the user.

5. Automates message handling to reduce the load on system administrators.

   Every message the comes to the console, from any system in the network, is automatically scanned by CA-Unicenter. Users set up rules telling Unicenter what to do when it sees any particular message. Based on the message, its source, and the time of day or day of week, CA-Unicenter will automatically take the action an administrator might have taken, from running another task to paging a programmer. This facility is of critical importance in a distributed environment because it can radically reduce the demand for additional system administrators.

6. Provides accountability for use and charge-back to recover costs.

   CA-Unicenter gathers all the UNIX accounting records and creates financial reports and user bills that reflect the organizational structure. It eases the burden of allocating overhead and making charge-back fair to all users.

7. Automates problem detection and help-desk management to improve service levels to your users and help ensure problems and promises don't get forgotten.

   CA-Unicenter offers a completely automated problem-ticket management system that monitors all open problem tickets and automatically escalates problem priority, or changes responsibility, when problems are not solved within selected time frames. In addition, the system automatically monitors the console log watching for pre-determined situations that should lead to open problem tickets. When it finds one, it automatically opens a problem ticket and monitors it until it is solved.

8. Manages report distribution to lower paper costs and reduce user frustration.

   CA-Unicenter can scan all printed reports, select the pages that should be delivered to individual users, and then deliver only those pages, either online or on paper.

9. Put the whole package in a graphical user interface that empowers users to do their own work.

   From security, to file management, to automated operations, and more, CA-Unicenter offers push-button GUI controls that limit the systems management knowledge required of users and eases the burden on central administration.

With the entire distributed processing industry backing Unicenter, and versions being released for new platforms every two months, user can be confident that their entire network of systems can be managed as a cohesive whole.

CA-Unicenter is a unique tool. Since its first availability in late 1992, hundreds of organizations have come to depend on it to manage their distributed systems.

For more information:

Computer Associates
703-709-4665

Paper Number - 4031

Title: *Object Database and 4Gls A Paradigm Shift or a Paradigm Transition?*

Information Builders, Inc.
1250 Broadway
New York, NY 10001
(212)736-4433

Copyright 1993 Information Builders, Inc.

Introduction

Object technology and object oriented database management systems (ODBMS) represent the next generation of technology whose time may be nearer than you think. By offering the promise of getting through the application backlog with code reusability, increase productivity, higher quality, and graphical access to information, many companies have already started pilot projects.

This paper examines the integration of object database technology with proven 4GL technology. Using HP's OpenODB and Information Builders FOCUS 4GL as the model we will learn if this integration represents a paradigm shift or a paradigm transition.

Specific topics include:

- What are objects and object database management systems?
- Comparison between HP's OpenODB and traditional object databases
- Why is OpenODB considered the next generation of technology?
- How can a 4GL enhance object database management systems applications?
- An application example: OpenODB with a 4GL (FOCUS)

Keywords: Hewlett-Packard Open Object Database - HP OpenODB
Information Builders, Inc., FOCUS - IBI 4GL
Information Builders, Inc., Enterprise Data Access/SQL - IBI Client/Server EDA/SQL
Database Management System - DBMS
Object-Oriented Database Management System - ODBMS
Object-Oriented SQL - OSQL
Relational Database Management System - RDBMS
Third Generation Language - 3GL
Fourth Generation Language - 4GL
Geographic Information System - GIS

Today's business environment presents increasing information management challenges such as growing demand for applications, integrating new technologies with existing technology, developing easier to use applications and accessing data regardless of locations.

Growing demand for applications:
There is a continued demand for new applications as well as for extending capabilities to existing ones. The application backlog problem has been around for many years, and improving programmer productivity is one way to address this problem. 4GLs have addressed the backlog problem to a large extent by providing programmer tools that increases productivity 10 to 1 over traditional 3GLs, yet there is still a crying need to reduce the application backlog. One reason for the every increasing backlog is that since 4GLs provide the tools to deliver applications quicker, the long term built-up demand for new applications or requests for application enhancements are coming forth at an even faster rate.

Integrating new technologies with existing technology:
The ability to integrate new technology while leveraging customer's current investment in existing technologies and employee knowledge is paramount. It is not cost effective nor acceptable in today's competitive environment to introduce a new technology in which you can not leverage existing knowledge. Training and acquisition of knowledge is expensive both in terms of money and resource. New technology must be able to integrate into the existent knowledge-base otherwise in most cases the introduction of new technology will not be acceptable.

Develop easier to use applications:
Customers continue to state the growing importance of end-user easy to use applications because it has such a tremendous impact on productivity. The combination of user friendly end-user tools of 4GLs combined with the reusability of objects provide the tools for the development of easy to use applications.

Accessing data regardless of location:
The need is becoming increasingly critical as companies continue to integrate their enterprise-wide information management systems to support all groups within an organization. This trend makes the requirement of accessing data on all platforms and

locations paramount to companies success today and in the future. Thus the capability of objects accessibility and availability across platforms is critical.

Now that we have stated the requirements of today's business environment let's begin the discussion of object database management systems.

What are objects and object database management systems?

Objects are computer simulated business information. The idea is to directly represent your business information in the computer, making it easier to track and manage your most valuable information. An object can by anything valuable you would find around you. For example a person, a process, or an item such as a telephone are examples of an object. An object is comprised of both data and the functions that are normally associated in a business object. Using an example of the object, Person, the object Person's data can be represented as Name, Address and Business Phone Number. Functions of the object person can include: Hire.

An **Object Database Management System (ODBMS)** is made up of objects combined with a database manager. An ODBMS provides capabilities to: share the same business information among many people, protect and manage the information, find information (no matter where it may be), and manage the entire business process (from beginning to the end of the process).

To determine the feasibility of incorporating ODBMS technology in your business, let's look at the some of the benefits of an ODBMS:

- Bring new services to market faster.
  Access to all information quickly and efficiently enable business decisions to be quickly and efficiently.

- Enable different applications to work together
  The sharing of objects across applications enable diverse applications to work together.

- Create more robust systems
  Systems can be assembled from proven components (reusability) which result in improved quality and reduced maintenance costs.

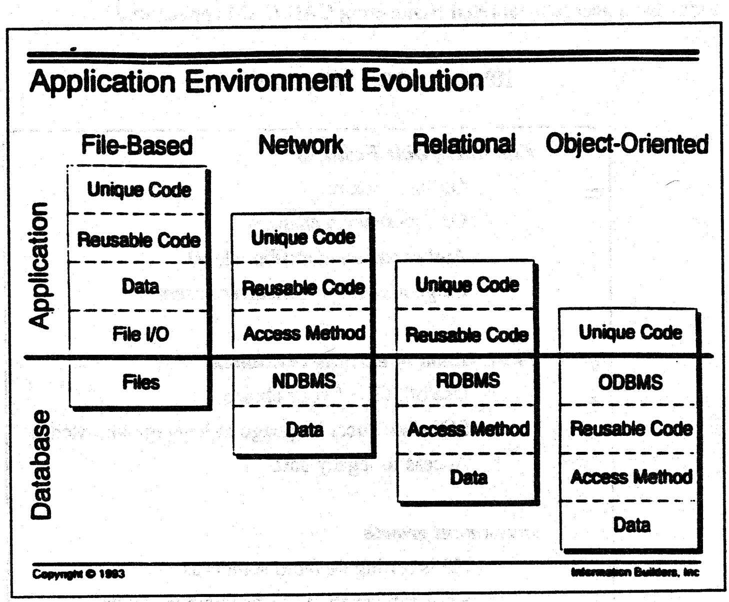
In addition to the benefits of ODBMS described above, ODBMS offers a faster and more efficient application development environment. Let's take a look at the application environment evolution so we can see why an ODBMS provides a more expedient methodology for rapid application development.

Flat files:	All work is handled by the application. Applications were designed to share data. All data was saved in files. This gave rise for a need to structure and share the data.
NDBMS: Network DataBase Management System	Network-model databases (including hierarchical) provide a common structure where data can be accessed by multiple applications and multiple users. However, each access method is unique making it difficult to share between applications and transfer knowledge. This gave rise to the need for an industry standard access method.
RDBMS: Relational DataBase Management System	Relational databases took some of the uniqueness out of the access method by providing an industry standard called SQL. SQL provides an ad-hoc request capability for data. The only thing lacking is a structure that represents the relationship between data.

ODBMS: Object-oriented databases combine the ad-hoc query capability found in RDBMS and the structure from NDBMS. It also includes the object modeling capabilities and the ability to reuse code.

Object
DataBase
Management
System

A graphic representation of the application environment evolution can be viewed in Example 1 below.


Application Environment Evolution
File-Based
Network
Relational
Object-Oriented
Application
Database
Unique Code
Reusable Code
Data
File I/O
Files
Unique Code
Reusable Code
Access Method
NDBMS
Data
Unique Code
Reusable Code
RDBMS
Access Method
Data
Unique Code
ODBMS
Reusable Code
Access Method
Data
Copyright © 1993
Information Builders, Inc

Example 1

As you can see from the above discussion, the application development environment has evolved so that more work is done in the Data Management System (DBMS). Objects, such as user-defined functions and types are stored as reusable code managed by the DBMS. The application programmer can then concentrate on program logic (application unique code) making applications less complex and thereby speeding up application development.

Comparison between HP's OpenODB and traditional object databases

There are several object database management systems (ODBMS) available in the market today. However, there is only one ODBMS, HP's OpenODB designed for commercial, enterprise-wide use (see Example 2). HP's OpenODB brings the customer forward to object technology without giving up key database management system (DBMS) features such as security, ad-hoc query language and database recovery. OpenODB is targeted at complex, commercial applications whereas current ODBMSs available on the market today have generally targeted engineering CAD/CAM applications.

HP's Open ODB Differentiators

Enterprise-wide Features
- On-line backup
- On-line schema changes
- Authorization at attribute level
- Large number of concurrent users

Preservation of existing investment
- Use of 3GL of your choice
- SQL-like query language to leverage knowledge
- Access to legacy data

Investment growth
- HP is setting de facto standards
- Research capability to maintain leadership

Mature software
- Based on established relational technology

Example 2

OpenODB protects your investment in technology by giving you the ability to incorporate existing 3GL programs into your ODBMS as well as providing an SQL-like query language to access data. HP is an industry leader in setting defacto standards and is currently working with the American National Standards Institute (ANSI) and the Object Management Group (OMG) to make HP's OSQL the industry standard. OpenODB is based on HP's relational technology, Allbase/SQL, which adds significant benefits including on-line backup, and on-line schema changes, and based on TPC results is the best performing relational database in the marketplace today.

One additional advantage of HP's OpenODB over other object-oriented DBMS is OpenODB's facility to store reusable code with the data in a shared environment. Other object oriented DBMS stores unique software and reusable code in the application domain. Therefore, using OpenODB technology enables the application unique software to remain separate and only the application specific code for applications must be developed.

Why is OpenODB considered to be the next generation of technology?

OpenODB is considered the next generation of technology because it builds on existing technology and also offers the object capabilities that bring forward application developers to the next generation of object technology. Let's look at the way OpenODB has been architected.

OpenODB builds on the developer's knowledge of SQL.
OSQL (Object Structured Query Language) is OpenODB's extension to SQL providing the ability for the user to define, manage, and query objects. By taking current SQL commands such as SELECT, OSQL adds the capability of objects.

All legacy applications and data are accessible.
OSQL provides access and integration of existing applications. ODBMS can front-end existing code modules (such as C++ or COBOL) and all legacy data is accessible through Information Builders, Inc., EDA (Enterprise Data Access)/SQL. EDA/SQL is data access, client/server technology that provides the ability to access over 50 data structures across 35 platforms (including HP-UX and HP MPE/iX) using standard protocols such as TCP/IP and SNA. HP has incorporated EDA/SQL into OpenODB such that OpenODB

can now gain access to relational, hierarchical, network, and keyed access data enabling investment of existing databases and technologies.

Common programming languages and tools can be used.
C, Pascal, FORTRAN, COBOL, C++ 3GL languages as well as 4GLs (FOCUS from Information Builders, Inc.) offer complete decision support tools that front-end OpenODB data.

Computer hardware strengths are maximized.
OpenODB is architected to support large work groups (500 users) thus enabling current investment in hardware to be maximized. Other object databases such as Gemstone, ObjectBase, Object Store, Objectivity/DB and Ontos are targeted toward small work groups (up to 8 users). HP's OpenODB is designed to work with large work groups, runs on the high performance HP-UX systems, and thus provides a high performance object oriented solution.

There are three object-oriented database approaches available today.

1) Relational DBMS with object oriented extensions has the DBMS basics of multi-user, high availability, security, reliability but few of the object-oriented basics.

2) Object-oriented program language-based databases (e.g., C++, or Smalltalk) has the object-oriented basics including unlimited user-defined types, inheritance, referential integrity for code but not the complete commercial DBMS basics.

3) OpenODB has both the power of relational DBMS coupled with object-oriented constructs.

OpenODB meets the needs of complex applications such as Geographic Information Systems (GIS) and customer service. OpenODB supports GIS which models query complex information relative to, for example, locations on a map.
GIS as well as customer service applications are found in many industries, including Telecommunications, Manufacturing, Aerospace/Defense and Healthcare.

The Physician's Workstation is another example of an object-oriented application. The

Physician's Workstation integrates a variety of data formats from around a hospital so that a doctor can view the whole picture about a patient. Incorporation of a knowledge-base can act as a filter working with the patient's information.

How can a 4GL enhance object database management systems applications?

Up to this point, we have been concentrating on the discussion of object databases. Before we discuss how 4GL tools can benefit object database management system applications, let's look at the tools available with Open ODB..
HP's OpenODB provides a graphical browser, that enables the user to look at the metadata (data about the objects) as well as object-oriented SQL that provide for simple query. 3GL programs can be used to extract and report from the object data structures. However, what tools are available for end-users to easily take OpenODB's data and turn it into information? How can the end-user get information (reports, graphs) etc. from the objects? A 4GL with easy to use, end-user tools that allow for quick development of reports and graphs is the answer. As of this time, only one 4GL exists has been architected to work with HP's OpenODB and that is FOCUS from Information Builders, Inc.

FOCUS tools provides the front-end to HP's OpenODB product which enables end-users, without any knowledge of FOCUS's 4GL language, to quickly and easily develop simple to complex reports and graphs. For programmers, FOCUS provides additional tools to address very sophisticated application development requirements. Enterprise Data Access (EDA)/SQL provides a backend extender to OpenODB such that it allows applications access (in addition to OpenODB data) over 50 databases (relational, network, hierarchical, and keyed access) across 35 platforms. HP's industrial strength OpenODB coupled with FOCUS's comprehensive decision support tools as well as EDA/SQL's data access, client/server capabilities provides for a most robust object oriented applications environment.

Since there are over 1,000,000 users worldwide using FOCUS, OpenODB with FOCUS will protect investment by leveraging FOCUS expertise throughout your business entities.

A well known industry consultant company, the Gartner Group, gives an excellent endorsement to both HP's OpenODB and Information Builders products.

The Gartner Group on OpenODB:

Tony Percy, an analyst with the Gartner Group said

> "It's ahead of its time because it's based on a proven, mature database engine. most of the products called object-oriented databases have been built from scratch and really hasn't matured as far as recovery, integrity and multi-user support."
>
> *Workstation News*

The Gartner Group on Information Builders, Inc. (IBI):

Lynn Berg, an analyst with the Gartner Group said:

> "Information Builders has become the de facto standard... to access legacy databases."

Another analyst from the Gartner Group continues:

> "Being an 'anything-to-anything' provider is a complex undertaking... but we believe (IBI's) plan is sound."

An application example: OpenODB with a 4GL (FOCUS)

Now that we have discussed the benefits of using object technology along with 4GLs we will now look at a specific example using FOCUS and OpenODB. Before we begin this example, let's define some concepts and definitions.

OpenODB is based on four constructs: objects, types, functions, and formats:

Object:	Provides concepts or things with identity
Type:	Provides a mechanism to group similar objects together (like CLASS). A subtype/supertype is equivalent to SUBCLASS/SUPERCLASS.
Function:	Provides ways to associate attributes, relationships, and behavior with objects (equivalent to ATTRIBUTE and METHOD).
Formats:	Datatype

Aggregate object is a collection of objects such as:

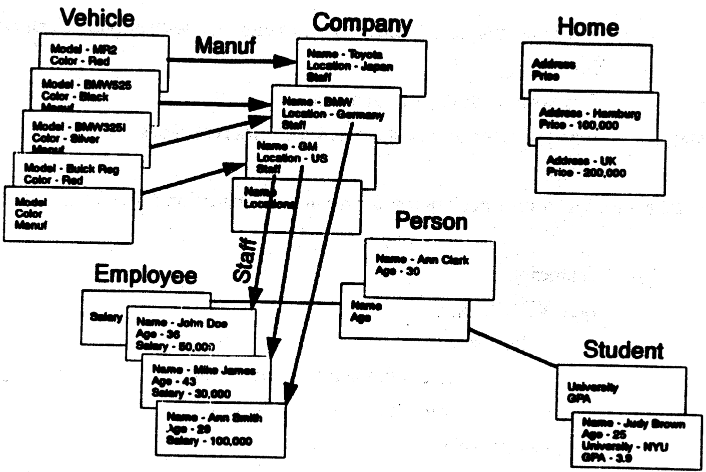
Bag:	Variable number of objects of the same type with duplicates allowed
List:	Sorted bag
Set:	Bag with no duplicates allowed
Tuple:	Fixed number of objects, may be of different types

If we take a look at a type declaration for a sample application it would like the following:

```
Type Declaration
        type VEHICLE
                functions:
                        model (vehicle) --> char (10)
                        color (vehicle) --> char (8)
        type COMPANY
                functions:
                        name (company) --> char (30)
                        locations (company) --> char (20)
        type PERSON
                functions:
                        name (person) --> char (20)
                        age (person) --> int
        type EMPLOYEE, subtype of PERSON
                functions:
                        salary (employee) --> double
```

A pictorial representing the functions with sample data can be found in Example 3.

FOCUS requires a mapping of the object data to the FOCUS engine. The two (2) files required to map the object data to the FOCUS engine are the Master File Description (MFD) and the Access File..


Vehicle
Model - MR2
Color - Red
Model - BMW525
Color - Black
Model - BMW325I
Color - Silver
Model - Buick Reg
Color - Red
Model
Color
Manuf
Manuf
Company
Name - Toyota
Location - Japan
Staff
Name - BMW
Location - Germany
Staff
Name - GM
Location - US
Staff
Name
Location
Home
Address
Price
Address - Hamburg
Price - 100,000
Address - UK
Price - 200,000
Staff
Employee
Salary
Name - John Doe
Age - 36
Salary - 50,000
Name - Mike James
Age - 43
Salary - 30,000
Name - Ann Smith
Age - 29
Salary - 100,000
Person
Name - Ann Clark
Age - 30
Name
Age
Student
University
GPA
Name - Judy Brown
Age - 25
University - NYU
GPA - 3.9

Functions

manuf (vehicle) -> company
staff (company) -> bagtype (employee)
owns (person, date) -> home
marriage (person -> bagtype (tupletype < char(20), date >)
children (person) -> bagtype (person)

Example 3

With FOCUS, the dictionary that maps the object data to FOCUS is called a Master File Description. The Access File is a complement to the Master File Description that determines what access path as well as other required information is stored. Both the Master File Description as well as the Access File are **automatically generated** for the user thus the user of FOCUS is not required to know the syntactical requirements of the FOCUS language.

Once the Master File Description and Access file are created the developer is done. From this point on, the end-user can create reports or graphs through FOCUS' end-user, point and click window driven, facilities. The end-user will automatically be directed through a series of windows where the user, for example, chooses fields, aggregation functions (such as sum., minimum, maximum, etc.) as well as choices of report formats including matrix reports. The FOCUS report language is automatically generated and users will have meaningful, easy to produce, reports from OpenODB data. FOCUS also enables the dynamic joining of heterogeneous data such as object, relational and network databases. Since OpenODB has been integrated with EDA/SQL, providing access to over 50 databases across 35 platforms, users would have the ability to access a joined structure that consists of, for example, OpenODB (from HP-UX), Image/SQL (from HP MPE/iX) and VSAM (from MVS). Therefore, reports can combine data from many different sources without the user knowing where the data is stored nor the access requirements of the various different database structures.

Summary

Flexible object database management systems coupled with 4GL technology enables you to leverage your investment in technology. HP's OpenODB is an industrial strength ODBMS that provides the ability to create reusable code providing for more expedient application development satisfying the ever growing demand for new and enhanced applications. FOCUS' easy to use decision support tools, offers OpenODB users the ability to turn data into meaningful information quickly and efficiently. OpenODB's EDA/SQL extender provides access to 35 heterogeneous data providing customers the ability to leverage investment in legacy data.

In conclusion, we believe the integrated ODBMS/4GL solution represents a paradigm transition that brings users forward into a new generation of technology. Users can maintain investment in existing applications by providing SQL like access as well as access to all data in the enterprise (through EDA/SQL). Users can continue to utilize proven 4GL tools such as FOCUS, and 3GLs such as C, Pascal, FORTRAN, COBOL and C++. Object technology is a passage into an new era of technology, a technology we perceive that is worthwhile investigating today and for the future.

For more information about HP's OpenODB contact, Hewlett-Packard Company, (408) 447-0000 (Doug Dedo or Cindi Nickol).

For more information about IBI's FOCUS or EDA/SQL products contact, Information Builders, Inc. (212) 736-4433 ext. 4320

Object Database and 4GLs:
A Paradigm Shift or a Paradigm Transition?

Michael Consoli
Information Builders

Copyright © 1993

DN 9600 643.0593

Object Database and 4GLs

Agenda

- What Is an Object Database?
- HP's OpenODB Versus Traditional ODBs
- Why Is ODB the Next Generation of Technology?
- How to Use Object Technology With Existing Applications
- How Can a 4GL Help You Meet ODB Needs?
- An Example: FOCUS With OpenODB

Copyright © 1993

DN 9600 643.0593

Object Database and 4GLs

Today's Business Environment

Information management challenges:

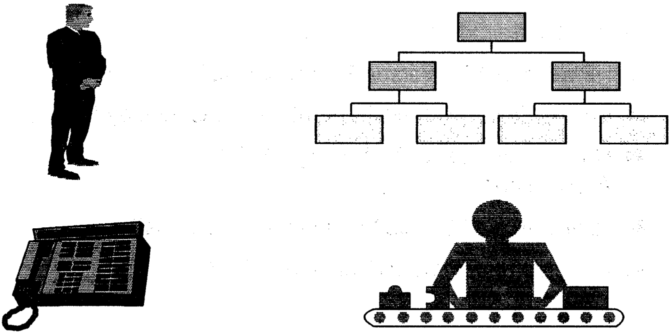
- Growing demand for applications
- Integrating new technologies with old
- Developing easier-to-use applications
- Accessing data regardless of location

The need to do more with less

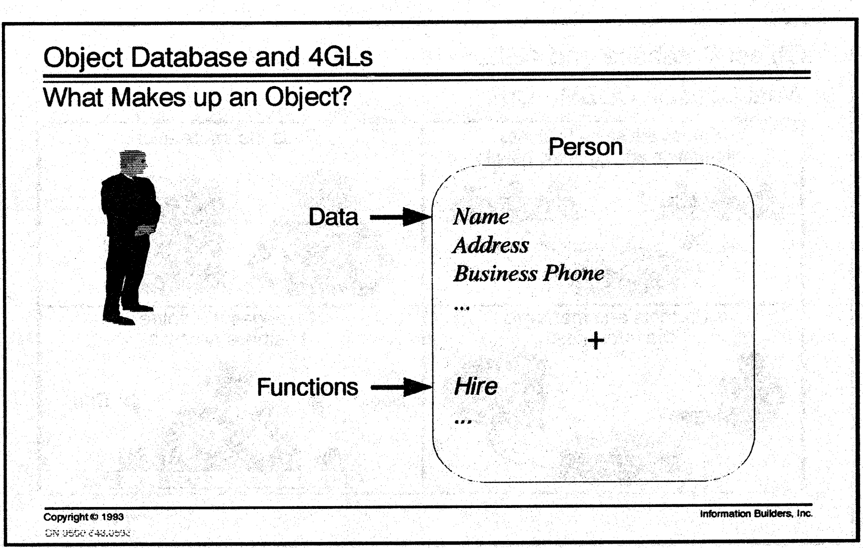
Copyright © 1993

Object Database and 4GLs

What Are Objects?



Copyright © 1993



Object Database and 4GLs

What Makes up an Object?

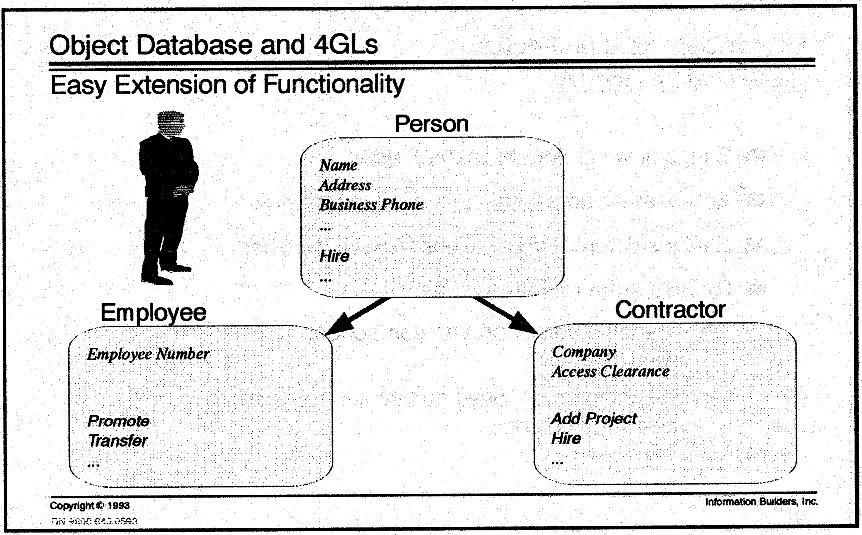
Person

Data → *Name*
Address
Business Phone
...

+

Functions → *Hire*
...

Copyright © 1993



Object Database and 4GLs

Easy Extension of Functionality

Person

Name
Address
Business Phone
...
Hire
...

Employee

Employee Number
Promote
Transfer
...

Contractor

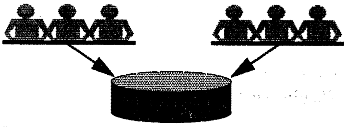
Company
Access Clearance
Add Project
Hire
...

Copyright © 1993

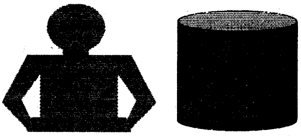
Object Database and 4GLs

What Does an ODBMS Do?

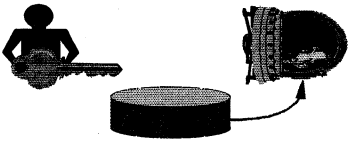
Shares the same business information among many people



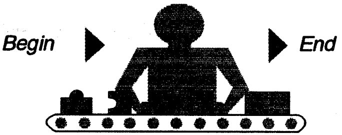
Finds the information



Protects and manages the information



Manages the entire business process



Copyright © 1993

DN 3500 643.0593

Information Builders, Inc.

Object Database and 4GLs

Benefits of an ODBMS

- Brings new services to market faster
- Accesses all data with a business model view
- Enables different applications to work together
- Creates more robust systems
  - Assembled from proven components (reusability)
  - Resulting in improved quality and reduced maintenance costs

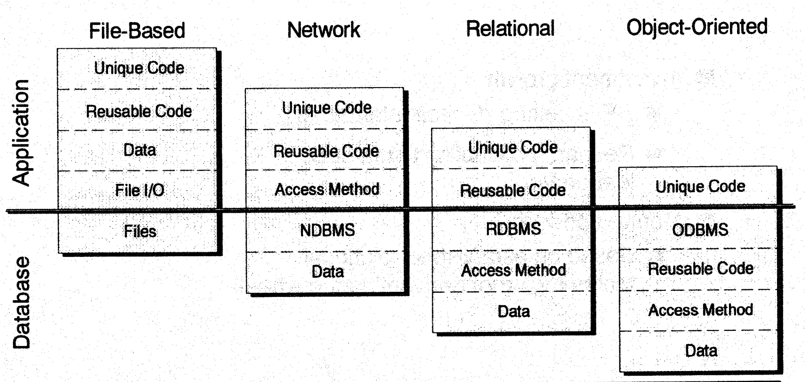
Copyright © 1993

DN 3500 643.0593

Information Builders, Inc.

Object Database and 4GLs

Application Environment Evolution



Application

Database

File-Based
Unique Code
Reusable Code
Data
File I/O
Files

Network
Unique Code
Reusable Code
Access Method
NDBMS
Data

Relational
Unique Code
Reusable Code
RDBMS
Access Method
Data

Object-Oriented
Unique Code
ODBMS
Reusable Code
Access Method
Data

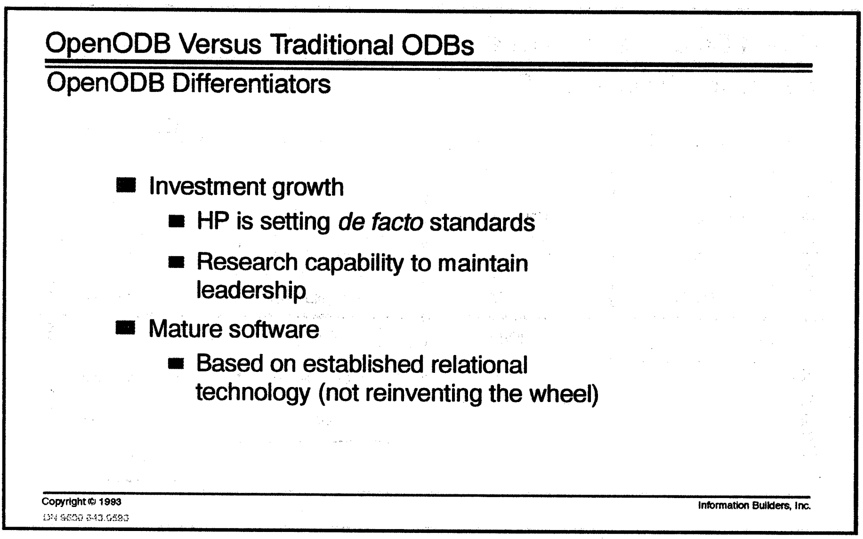
Copyright © 1993

OpenODB Versus Traditional ODBs

OpenODB Differentiators

- Enterprise-wide features
  - Online backup
  - Online schema changes
  - Authorization at attribute level
  - Large number of concurrent users
- Preservation of existing investment
  - Use the 3GL of your choice
  - SQL-like query language to leverage knowledge
  - Access to legacy data

Copyright © 1993



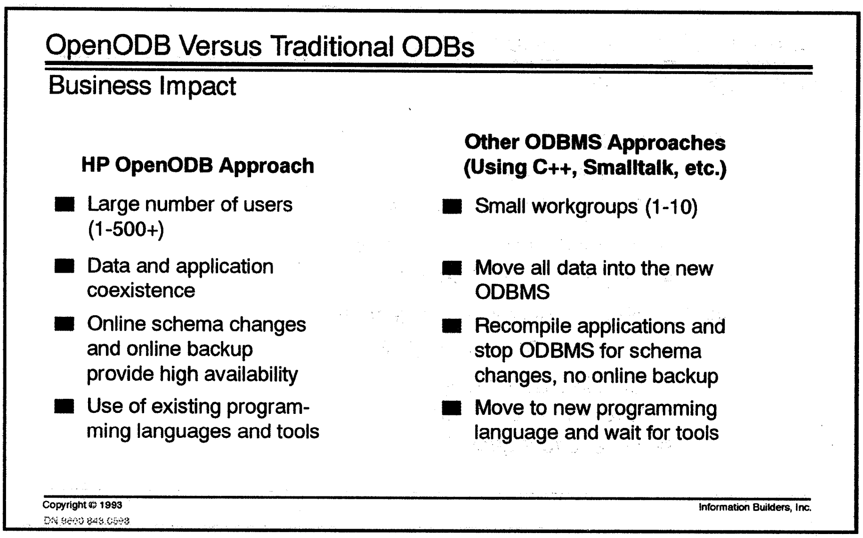
OpenODB Versus Traditional ODBs

OpenODB Differentiators

- Investment growth
  - HP is setting *de facto* standards
  - Research capability to maintain leadership
- Mature software
  - Based on established relational technology (not reinventing the wheel)

Copyright © 1993

Information Builders, Inc.



OpenODB Versus Traditional ODBs

Business Impact

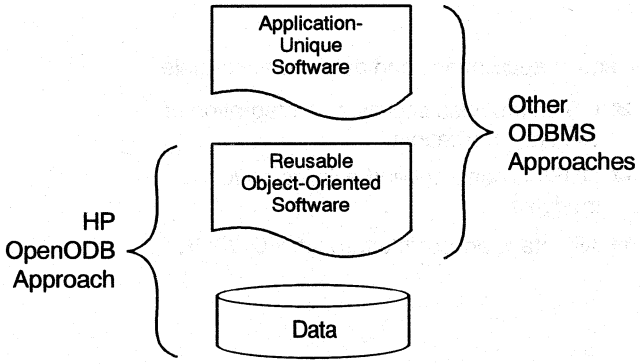
HP OpenODB Approach	**Other ODBMS Approaches (Using C++, Smalltalk, etc.)**
Large number of users (1-500+)	Small workgroups (1-10)
Data and application coexistence	Move all data into the new ODBMS
Online schema changes and online backup provide high availability	Recompile applications and stop ODBMS for schema changes, no online backup
Use of existing programming languages and tools	Move to new programming language and wait for tools

Copyright © 1993

Information Builders, Inc.

OpenODB Versus Traditional ODBs

Object-Oriented Applications


Application-Unique Software
Other ODBMS Approaches
Reusable Object-Oriented Software
HP OpenODB Approach
Data

Copyright © 1993

ODB: The Next Generation

Why Is OpenODB the Next Generation of Technology?

Evolutionary versus revolutionary

- Builds on the developer's knowledge of SQL
- OSQL – defines, manages, and queries objects
- Takes current SQL commands (ex. SELECT) and adds capability of objects

Copyright © 1993

ODB: The Next Generation

Why Is OpenODB the Next Generation of Technology?

All legacy applications and data are accessible

- OSQL provides access and integration of existing applications
- ODBMS can front end existing code modules
- All data is accessible through EDA/SQL

Copyright © 1993

ODB: The Next Generation

Why Is OpenODB the Next Generation of Technology?

Common programming languages and tools can be used:

- C, Pascal, FORTRAN, COBOL, C++
- 4GLs (FOCUS) for application development

Copyright © 1993

ODB: The Next Generation

Why Is OpenODB the Next Generation of Technology?

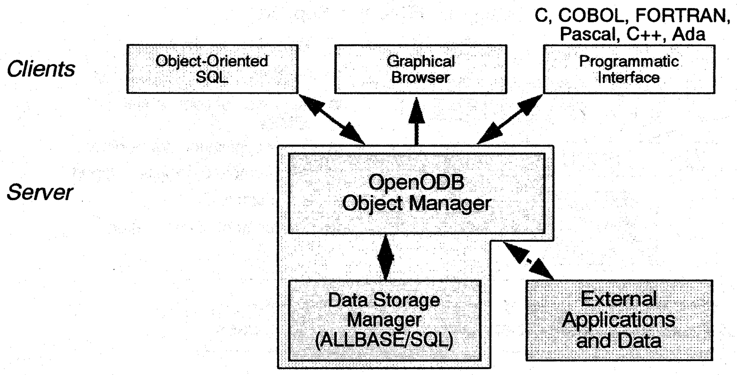
Computer hardware strengths are maximized:

- OpenODB architected to support many users
- Current investment in hardware can be maximized

Copyright © 1993

ODB: The Next Generation

OpenODB From Hewlett-Packard



Clients

Server

C, COBOL, FORTRAN, Pascal, C++, Ada

Object-Oriented SQL

Graphical Browser

Programmatic Interface

OpenODB Object Manager

Data Storage Manager (ALLBASE/SQL)

External Applications and Data

Copyright © 1993

Database Spectrum

1 — Number of Users — 500+

Relational

dBASE
R:BASE

ALLBASE/SQL
Informix
Ingres
Oracle
Sybase

Object-Oriented

Gemstone
ObjectBase
ObjectStore
Objectivity/DB
Ontos

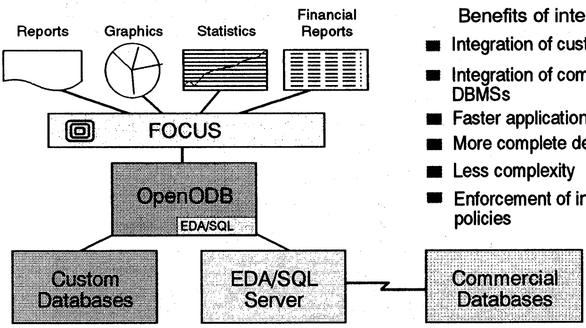
HP OpenODB

Copyright © 1993

How Can a 4GL Meet ODB Needs?

Information Builders and Hewlett-Packard

Integrated Decision Support


Reports
Graphics
Statistics
Financial Reports
FOCUS
OpenODB
EDA/SQL
Custom Databases
EDA/SQL Server
Commercial Databases

Benefits of integration

- Integration of custom DBMSs
- Integration of commercial DBMSs
- Faster application development
- More complete decision support
- Less complexity
- Enforcement of information policies

Copyright © 1993

How Can a 4GL Meet ODB Needs?

Leverage FOCUS Expertise

- All decision support tools available:
  - TABLE
  - Analyze
  - GRAPH
  - Dialogue Manager
  - Subroutine support
  - Talk technology
- Extensions in the future to handle multimedia data

Copyright © 1993

Aberdeen Group on Open ODB

Peter Kastner, an analyst with Aberdeen Group, agrees with HP's claims that its object-oriented database technology is two-to-three years ahead of the rest of the industry.

"HP made two very astute decisions," said Kastner. "One was to build its object-oriented database on top of and in conjunction with a relational database, which is going to ease the impact on thousands of users who are just gearing up for relational technology.... And they have designed it in such a way that it could work on virtually any relational platform, which is the open part of it."

— *The HP Chronicle*

Copyright © 1993

Gartner Group on OpenODB

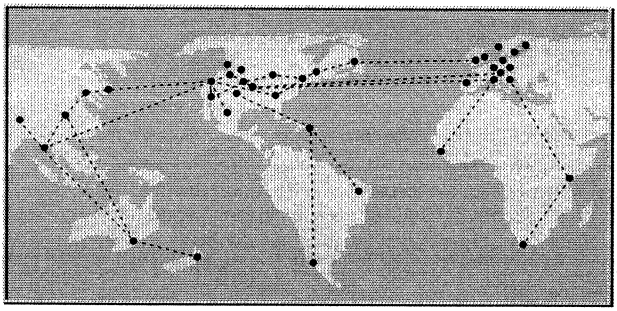
Tony Percy, an analyst with the Gartner Group, said, "It's ahead of its time because it's based on a proven, mature database engine. Most of the products called "object-oriented databases" have been built more from scratch and really haven't matured as far as recovery, integrity, and multiuser support."

— *Workstation*

Copyright © 1993

Using New Technology With Existing Applications

Geographic Information System

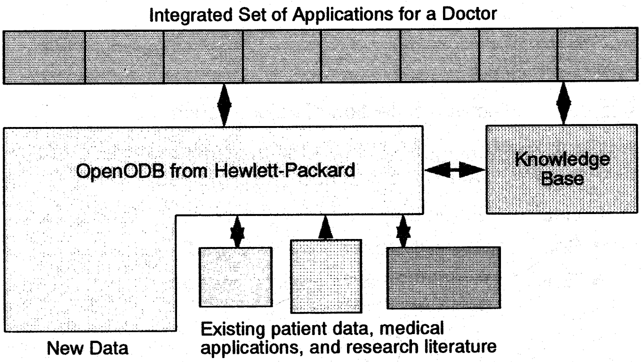


Model and query complex information relative to locations on a map

Copyright © 1993

Using New Technology With Existing Applications

Integrating Medical Information


Integrated Set of Applications for a Doctor
OpenODB from Hewlett-Packard
Knowledge Base
New Data
Existing patient data, medical applications, and research literature

Copyright © 1993

An Example

Concepts and Definitions

OpenODB is based on four constructs:
objects, types, functions, and **formats**

- Object: Provides concepts or things with identity
- Type: Provides a mechanism to group similar objects together (like CLASS).
  A subtype/supertype is equivalent to SUBCLASS/SUPERCLASS.
- Function: Provides ways to associate attributes, relationships, and behavior with objects (equivalent to ATTRIBUTE and METHOD)
- Format: Datatype

Copyright © 1993

An Example

Concepts and Definitions: Objects

OpenODB supports atomic and aggregate objects:

- Atomic object is indivisible
- Aggregate object is a collection of objects, such as:

Bag:	Variable number of objects of the same type with duplicates allowed
List:	Sorted bag
Set:	Bag with no duplicates allowed
Tuple:	Fixed number of objects, may be of different types

Copyright © 1993

An Example

Concepts and Definitions

- Types
    - OpenODB supports type inheritance: all objects belonging to a subtype belong to all its supertypes; a function defined on a supertype is also defined for all subtypes, but may be overriden. Multiple inheritance is also supported.
- Functions:
    - Functions are defined over types and constrain the behavior of objects. Functions in OpenODB are used to:
        1. Define an attribute
        2. Specify the relationships among objects
        3. Specify behavior of objects
    - Functions serve two purposes:
        1. To get the value of an attribute
        2. To establish a relationship between objects
- Formats
    - Decimal
    - Double
    - Real
    - Integer
    - SmallInt
    - Char (varlen)
    - Binary (varlen)
    - Boolean
    - Date
    - Time
    - DateTime
    - Interval

Copyright © 1993

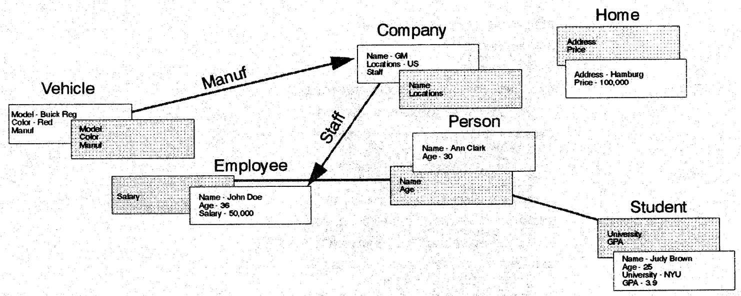
An Example

Type Declaration

```
type VEHICLE
    functions:
        model (vehicle) -> char(10)
        color (vehicle) -> char(8)
type COMPANY
    functions:
        name (company) -> char(30)
        locations (company) -> char(20)
type PERSON
    functions:
        name (person) -> char(20)
        age (person) -> int
type EMPLOYEE, subtype of PERSON
    functions:
        salary (employee) -> double
type STUDENT, subtype of PERSON
    functions:
        university (student) -> char(30)
        gpa (student) -> double
type HOME
    functions:
        address (home) -> char(30)
        price (home) -> double
```

Copyright © 1993

An Example

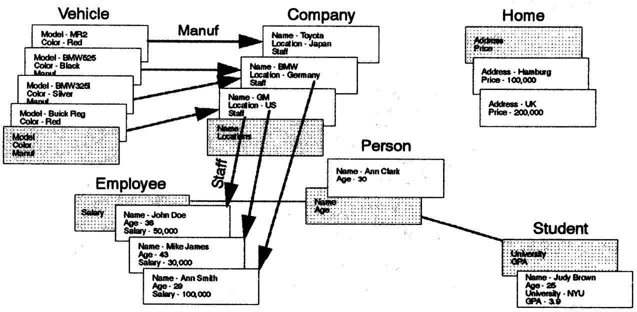

Home
Address
Price
Address - Hamburg
Price - 100,000
Company
Name - GM
Locations - US
Staff
Name
Locations
Vehicle
Manuf
Model - Buick Reg
Color - Red
Manuf
Model
Color
Manuf
Staff
Person
Name - Ann Clark
Age - 30
Name
Age
Employee
Salary
Name - John Doe
Age - 36
Salary - 50,000
Student
University
GPA
Name - Judy Brown
Age - 25
University - NYU
GPA - 3.9

Functions

manuf (vehicle) -> company
staff (company) -> bagtype (employee)
owns (person, date) -> home
marriage (person) -> bagtype (tupletype < char(20), date >)
children (person) -> bagtype (person)

Copyright © 1993

An Example


Vehicle
Model - MR2
Color - Red
Model - BMW525
Color - Black
Manuf
Model - BMW325I
Color - Silver
Manuf
Model - Buick Reg
Color - Red
Model
Color
Manuf
Manuf
Company
Name - Toyota
Location - Japan
Staff
Name - BMW
Location - Germany
Staff
Name - GM
Location - US
Staff
Name
Locations
Home
Address
Price
Address - Hamburg
Price - 100,000
Address - UK
Price - 200,000
Staff
Employee
Salary
Name - John Doe
Age - 36
Salary - 50,000
Name - Mike James
Age - 43
Salary - 30,000
Name - Ann Smith
Age - 29
Salary - 100,000
Person
Name - Ann Clark
Age - 30
Name
Age
Student
University
GPA
Name - Judy Brown
Age - 25
University - NYU
GPA - 3.9

Functions

manuf (vehicle) -> company
staff (company) -> bagtype (employee)
owns (person, date) -> home
marriage (person -> bagtype (tupletype < char(20), date >)
children (person) -> bagtype (person)

Copyright © 1993

Corresponding Master File

```
FILE=VEHICLE, SUFFIX=ODB, $

SEGMENT=VEHICLE
   FIELD=MODEL,, A10, A10, $
   FIELD=COLOR,, A8, A8, $

SEGMENT=COMPANY, PARENT=VEHICLE
   FIELD=NAME,, A30, A30, $
   FIELD=LOCATIONS, A20, A20, $

SEGMENT=PERSON, PARENT=COMPANY
   FIELD=NAME,, A20, A20, $
   FIELD=AGE,, I4, I4, $
   FIELD=SALARY,, D6.2, D6.2, $
   FIELD=UNIVERSITY,, A30, A30, $
   FIELD=GPA,, D6.2, D6.2,$
   FIELD=PERSEGTYPE,, A10,, $

SEGMENT=MARRIAGE, PARENT=PERSON
   FIELD=MARRIAGE_C,, A20, A20, $
   FIELD=MARRIAGE_D,, YMD, YMD, $

SEGMENT=CHILDREN, PARENT=PERSON
   FIELD=NAME,, A20, A20, $
   FIELD=AGE,, I4, I4, $
   FIELD=SALARY,, D6.2, D6.2, $
   FIELD=UNIVERSITY,, A30, A30, $
   FIELD=GPA,, D6.2, D6.2,$
   FIELD=PERSEGTYPE,, A10,, $

SEGMENT=HDATE, PARENT=PERSON
   FIELD=HD,, YMD, YMD, $

SEGMENT=HOME, PARENT=HDATE
   FIELD=ADDR,, A30, A30, $
   FIELD=PRICE,, D8.2, D8.2, $
```

Copyright © 1993

Corresponding Master File

```
FILE=VEHICLE, SUFFIX=ODB, $

SEGMENT=VEHICLE
   FIELD=MODEL,, A10, A10, $
   FIELD=COLOR,, A8, A8, $

SEGMENT=COMPANY, PARENT=VEHICLE
   FIELD=NAME,, A30, A30, $
   FIELD=LOCATIONS, A20, A20, $

SEGMENT=PERSON, PARENT=COMPANY
   FIELD=NAME,, A20, A20, $
   FIELD=AGE,, I4, I4, $
   FIELD=SALARY,, D6.2, D6.2, $
   FIELD=UNIVERSITY,, A30, A30, $
   FIELD=GPA,, D6.2, D6.2,$
   FIELD=PERSEGTYPE,, A10,, $
```

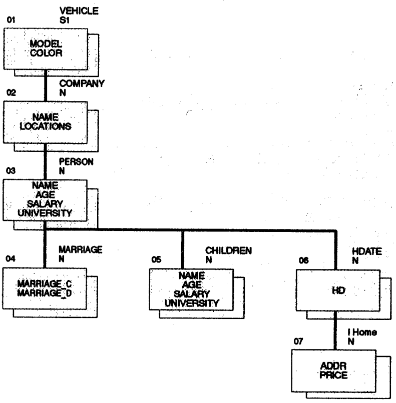
Copyright © 1993

Corresponding Master File

```
SEGMENT=MARRIAGE, PARENT=PERSON
   FIELD=MARRIAGE_C,, A20, A20, $
   FIELD=MARRIAGE_D,, YMD, YMD, $

SEGMENT=CHILDREN, PARENT=PERSON
   FIELD=NAME,, A20, A20, $
   FIELD=AGE,, I4, I4, $
   FIELD=SALARY,, D6.2, D6.2, $
   FIELD=UNIVERSITY,, A30, A30, $
   FIELD=GPA,, D6.2, D6.2,$
   FIELD=PERSEGTYPE,, A10,, $

SEGMENT=HDATE, PARENT=PERSON
   FIELD=HD,, YMD, YMD, $

SEGMENT=HOME, PARENT=HDATE
   FIELD=ADDR,, A30, A30, $
   FIELD=PRICE,, D8.2, D8.2, $
```

Copyright © 1993

Structure of OpenODB


01
VEHICLE
S1
MODEL
COLOR
02
COMPANY
N
NAME
LOCATIONS
03
PERSON
N
NAME
AGE
SALARY
UNIVERSITY
04
MARRIAGE
N
MARRIAGE_C
MARRIAGE_D
05
CHILDREN
N
NAME
AGE
SALARY
UNIVERSITY
06
HDATE
N
HD
07
I Home
N
ADDR
PRICE

Copyright © 1993
DN 9600 243.0593
Information Builders, Inc.

Example: Table Requests

Retrieve name and age of all students

```
TABLE FILE ODBSSAM
PRINT NAME AGE
END
```

Result:

```
NAME                               AGE
Ann Clark                           30
Ann Smith                           29
John Doe                            36
Judy Brown                          25
Mike James                          43
```

Retrieve name and age of all students

```
TABLE FILE ODBSSAM
PRINT NAME AGE
WHERE PERSEGTYPE EQ STUDENT;
END
```

Result:

```
NAME                               AGE
Judy Brown                          25
```

Copyright © 1993
DN 9600 243.0593
Information Builders, Inc.

Example: Table Requests

Retrieve amount of money earned by all working people

```
TABLE FILE PEOPLE
SUM SALARY
END
```

Result:

```
SALARY
180,000
```

Retrieve all schools

```
TABLE FILE ODBSSAM
PRINT UNIVERSITY
END
```

Result:

```
UNIVERSITY
NYU
```

Copyright © 1993

Example: Table Requests

Retrieve all schools and companies

```
TABLE FILE ODBSSAM
PRINT UNIVERSITY COMPANY BY NAME
END
```

Result:

```
NAME              SCHOOL       COMPANY
Ann Clark           .            .
Ann Smith           .          BMW
John Doe            .          GM
Judy Brown        NYU            .
Mike James          .          GM
```

Retrieve names and ages of all children of all employees of GM

```
TABLE FILE ODBSSAM
PRINT CHILDREN.NAME CHILDREN.AGE
BY NAME
WHERE COMPANY IS 'GM' AND
      PERSEGTYPE IS 'EMPLOYEE';
END
```

Result:

```
NAME           CHILDREN.NAME       CHILDREN.AGE
Ann Clark      Judy Brown          25
```

Copyright © 1993

Example: Table Requests

Retrieve address and prices of homes
for all people employed by BMW

```
TABLE FILE ODBSSAM
PRINT ADDR PRICE
BY NAME
WHERE COMPANY IS 'BMW' AND
      PERSEGTYPE IS "EMPLOYEE"
END
```

```
Result:
    NAME              Address         Price
    Ann Smith         Hamburg         100,000
```

Copyright © 1993
DN 9800 043.0593
Information Builders, Inc.

ODB: the Next Generation

OpenODB – a Paradigm Transition

- Maintain investment in existing applications
  - Superset of SQL (OOSQL)
  - Access to data in the enterprise through EDA/SQL
- Utilize proven 4GL tools and languages
  - FOCUS 4GL
  - C, Pascal, FORTRAN, COBOL, C++
- Support commercial (large number of users) applications

Copyright © 1993
DN 9800 043.0593
Information Builders, Inc.

Paper #4032

Implementing HP 9000-890 Platforms:

A Case Study

INTEREX Conference

September 1993

Samuel C. Ellis, Ph. D.
Portland Community College
Portland, Oregon 97219
(503) 244-6111

Implementing HP 9000-890 Platforms:

A Case Study

1. Abstract:

Portland Community College recently acquired two HP 9000-890 "Emerald" platforms. The goal of this paper is to outline the critical issues that determined the success of the implementation. There will be a discussion of important factors that should considered by those who install such systems.

PCC operated in a centralized environment with a Honeywell-Bull mainframe for many years. All application software was developed internally. The college released an RFI and RFP for new hardware and software in 1992; Hewlett-Packard, NCR, Sequent, IBM, DEC, Honeywell-Bull, and others submitted proposals. Each vendor suggested different strategic directions for the college. For example, there were a variety of approaches to data communications and networking. As another example, the proposed machines involved different hardware architectures.

This paper focuses on three important topics:

a. Portland Community College's strategic vision.

b. A description of how well Hewlett-Packard shared PCC's vision.

c. The process of implementing the vision.

PCC and Hewlett-Packard have confronted many issues -- right-sizing, application porting, staff training, machine performance, ancillary system software, data communications, hardware sizing and configuration, technical support, machine architecture and engineering, open systems technology, and partnerships. The implementation has involved many successes. This paper describes those situations so that others can prepare effective implementation plans.

2. Description of Portland Community College:

PCC is a two year institution with five major campuses and a number of satellite locations. It offers programs in a metropolitan area covering 1,500 square miles. The college employs approximately 3,500 people and serves 90,000 students. PCC has an annual budget of over $100,000,000 from a variety of sources.

Until recently, the information services department operated in a centralized fashion with a Honeywell-Bull mainframe. All applications were developed in-house. Almost all support, training, product development, operations, etc., were concentrated at a central site.

3. PCC's Strategic Vision:

In 1990 the college sensed that it could no longer be effective with its existing computing efforts. A consultant was hired to document the difficulties and suggest changes. After a year of careful study the consultant concluded that significant improvements were needed in hardware, software, customer service, and operations. Lengthy reports were presented to the president and Board of Directors about overhauling the college's computing activities.

One of the consultant's recommendations involved the hiring of an associate vice president who would direct administrative computing, instructional computing, networking, and telephone services. The position was filled in September 1991; PCC began an aggressive program to improve all aspects of its computing.

a. RFI/RFP Process:

Within a few weeks the college began a Request for Information (RFI) and Request for Proposal (RFP) process in order to identify new hardware and software for administrative computing. PCC wrote lengthy documents that asked vendors to propose products and services that would meet the college's needs. There were several opportunities for vendors to demonstrate their systems. In addition, PCC visited several customer sites to evaluate hardware and software.

There were two RFI/RFP processes that ran parallel with each other -- one for hardware and one for software. The goal was to leverage many potential combinations of products and services; it was important to keep costs to a minimum.

Several strategic directions emerged during the RFI/RFP process. PCC asked vendors to explain how their products and services addressed nine important areas:

- Distributed computing and client-server architecture.
- User ownership and responsibility for certain parts of the systems.
- Open systems, interoperability, portability, and flexibility.
- Networking, data communications, and the development of an information utility.
- Enhanced customer service.
- Partnerships versus simple sales relationships.
- Graphical user interfaces.
- Multi-processor environments and application-specific servers.
- Relational data base management systems.

RFP proposals were evaluated with the following criteria:

- Functionality.
- Cost.
- Product integration.
- Vendor's qualifications.
- System performance.
- Installation methodology.
- Consistency with the college's strategic directions.
- Technical support.
- Product system life.
- Responsiveness to the RFP.
- Partnerships.

b. Technical Currency:

Many RFPs that are issued by public agencies require proposed hardware to be operational at several customer sites. PCC had no such requirement; the college encouraged companies to suggest future technologies. The goal was to invest in systems that would have relatively long life. PCC felt that installing "status quo" hardware would be short-sighted. In addition to the desire for future technology, the college was aggressive in its pursuit of how those systems could be upgraded.

c. Open Systems:

There was substantial emphasis placed on the college's desire for open systems. PCC declared that it would not acquire products that involved proprietary technology. The RFP gave a precise definition and philosophical description about open systems.

d. Partnerships:

It was the college's sense that the term "partnership" was being used by many people but that it had very little meaning. Vendors used the term as part of their marketing presentations; colleges sometimes spoke of partnerships as a euphemism for donations. The college consistently challenged companies to cut through the rhetoric; it was important to be specific about how PCC and its chosen vendors would ensure wide-scale success.

e. Information Utility:

Administrative computing hardware and software should fit within an overall plan for information resources. Specifically, the college was convinced that it needed to anticipate the requirements of future applications and develop a supporting infrastructure. Vendors were asked how their proposed products would be consistent with an enterprise-wide information utility -- the convergence of voice, data, and video systems at the desktop.

PCC was committed to the development of a high performance infrastructure. The college predicted that ATM would replace FDDI, multimedia applications would expand significantly, distributed computing and client-server architectures would grow, decentralization would increase, and users would have extraordinary connectivity requirements. Each desktop would need a standard information utility outlet to provide access to all desired information resources.

f. The Sandbox:

One of the college's most important efforts involved the development of "The Sandbox." The Sandbox is a tool for creative problem-solving and negotiation. Ordinary RFP processes tend to have unrealistic goals and involve adversarial relationships. In spite of well-defined specifications, RFPs fail to address a number of important issues. In addition, there is a reliance upon liquidated damages and performance bonds instead of constructive resolution of problems.

Many vendors were unable to grasp the meaning of the Sandbox principle. They felt more comfortable soliciting information about the college's specifications, evaluation criteria, and decision process for awarding bids. Their interest was in issuing quotes, responding to a list of system requirements, and making a sale. PCC had little interest in such narrow approaches to doing business.

The Sandbox principle can be described in two ways. First, the customer invites vendors to propose solutions without the vendors knowing some of the basic problems. Companies are asked to propose a variety of products and services that would be "ideal" for a progressive institution. Such a backwards arrangement puts a heavy burden on vendors; they become responsible for defining systems. PCC was able to react to the initiatives of the vendors and not vice-versa. Such a situation puts the customer in a position of strength; it is sometimes advantageous to be approached rather than approaching others.

Second, the Sandbox allowed PCC to establish firm boundaries -- items on which there could be no compromise (e.g., maximum budget, technology direction, performance expectations). Vendors were then able to suggest how the remaining variables could be adjusted in order to achieve success. The goal of the Sandbox is to create flexibility and maneuverability. It is unlikely that an effective implementation will occur if there is too much reliance on specifications and requirements. Instead, there should be firm boundaries, trust, cooperation, and a willingness to solve problems creatively within the boundaries.

g. Mission Criticality:

PCC's applications are highly critical. The failure of any system would be an extraordinary problem. It would be a serious situation, for example, to have a failure during student registration. Thousands of students would be denied service; the college's revenue would be harmed.

Hewlett-Packard has a reputation for engineering highly reliable products. The mean-time-between failure (MTBF) is significant. In addition, customers have a variety of options for disc mirroring, fault tolerance, and switchover. PCC feels absolutely confident about the protection of its mission critical applications.

4. Hewlett-Packard and PCC's Strategic Vision:

The relationship between Hewlett-Packard and PCC got off to a slow start. There were a number of organizational changes in Hewlett-Packard; no sales representative was assigned to the college to discuss PCC's needs.

Finally, though, Hewlett-Packard put together an excellent team that handled the RFI and RFP processes very well. They listened carefully and were able to adapt to the Sandbox methodology. The systems engineers did an outstanding job of proposing systems, refining requirements, and ensuring a suitable hardware environment. There were numerous "beat it up" sessions during which we reviewed various equipment configurations. It was gratifying that such meetings were not sales-oriented; Hewlett-Packard's focus was on trying to ensure a superior operating environment.

The RFP process included a visit to Cupertino, California to discuss HP 9000-890 technology. It was helpful to learn about hardware architecture, machine performance, upgrade paths, and open systems strategies. There was substantial discussion about running sophisticated Oracle applications on Emerald systems.

At times Hewlett-Packard drifted into marketing rhetoric; they were conditioned to present reports about the fine qualities of their company. Those moments were brief, though, and there was a prompt return to concrete issues.

It was tiresome to hear all of the vendors, including Hewlett-Packard, make substantial claims about their technical superiority. Every company argued that they had the highest TPC rating, closest working relationship with Oracle, and best endorsement from the Gartner Group. The college was not impressed with such discussions.

New customers should consider attending the Cambridge Conference on Open Systems that is held several times per year in Massachusetts. Dr. John Donovan makes an extraordinary presentation for executives about strategic directions. Many participants consider it one of the best seminars they have ever experienced. Most Hewlett-Packard sales representatives are aware of the conference and can make arrangements for customers to attend.

Hewlett-Packard did an outstanding job with respect to the college's strategic vision. They addressed all of the RFP specifications, demonstrated leadership in open systems, showed that the HP 9000-890s would support PCC's application software, and provided technical support to ensure success. In addition, Hewlett-Packard eliminated many of the college's fears. It wasn't necessary for PCC to worry about RISC technology (in contrast with DEC Alpha), adequate capacity (in contrast with IBM RS 6000), etc. Hewlett-Packard managed to reduce risk, increase performance, and maintain affordability. Doing business with Hewlett-Packard was an easy decision. Perhaps the most important criterion, though, was the trust that was developed. It always seemed that Hewlett-Packard cared about the college and was committed to its success. They understood how PCC wanted to do business and found a way to enhance the college's efforts.

PCC eventually ordered a one-way HP 9000-890, two-way HP 9000-890, 45 gigabytes of disc, DAT tape drives, line printers, data communications equipment, consulting services, and system software. Later, the college bought several additional small business servers for small applications. PCC has spent approximately $1,700,000 on hardware, system software, premium technical support, and training.

The two-way processor is used for the college's SCT Banner administrative computing software -- accounting, payroll, student registration, financial aid, etc. SCT Banner is based upon an Oracle relational data base management system. The two-way processor supports 500 concurrent users without any difficulty. The one-way processor is used for Oracle Mail, Dynix library functions, room scheduling, and development projects. CA-UNICENTER is used for system management across all platforms.

5. Implementation:

There are several implementation issues that should considered by anyone who installs HP 9000-890 platforms. Hewlett-Packard did an outstanding job of addressing those topics.

a. Premium Support and Wraparound Services:

In spite of PCC's careful planning and capable staff, there were many issues that the college was unable to handle. There were significant technical challenges that emerged during the implementation. Customers need to seek premium support and wraparound services from Hewlett-Packard in order to be successful. Such arrangements should be made during contract negotiations and bundled with the cost of the systems. It is advisable to purchase services for a flat fee rather than a time and materials basis.

Hewlett-Packard did an excellent job of insulating the college from many problems. And, those who provided technical support helped PCC's staff learn how to resolve difficulties. There was an effort to make the information services department self-sufficient rather than dependent upon vendors.

The cost of premium support and wraparound services was quite justified. The college avoided substantial frustration and wasted effort. We were able to maintain our conversion schedule and transition out of the Honeywell-Bull environment. In addition, Hewlett-Packard's support helped us remain effective with our customers; we knew how to implement systems and provide good service. The information services department was able to make steady progress, keep a positive perspective, and generate enthusiasm for the implementation. Our project would not have been successful without Hewlett-Packard's assistance.

b. Loaner Equipment:

The college made its decision to purchase HP 9000-890 platforms in June 1992. The machines, though, were not scheduled for shipment until December 1992. Hewlett-Packard solved the problem by providing loaner equipment, an HP 9000-867. The arrangement was quite successful; performance was good, the application software worked well, and training could begin immediately.

c. Training:

The movement away from a mainframe environment to Unix systems is a significant endeavor. The information services staff had no experience in many of the areas in which they would be working. There was some apprehension, uncertainty, and fear. It was important to provide an effective training plan -- not only for technical reasons, but so that the staff could adjust to new ways of doing business.

Before sending anyone to Hewlett-Packard training the staff attended a series of informal workshops. One of the college's instructors was hired to talk about Unix concepts; another person provided information about Oracle. The goal was to help everyone develop a frame of reference for their future training. PCC sensed that it was necessary to ease the staff into an understanding of the new systems; it was not appropriate to immediately thrust them into unfamiliar territory. We wanted people to be prepared for the formal training that they would receive.

Unix manuals, on-line tutorials, and documentation were made available to the staff. A technical library was built so that there would be a quiet place to study. In addition, we solicited concerns and questions in advance of training so that the instructor would be prepared to handle important topics.

All of the staff gave positive reports about Hewlett-Packard's training. Some described it as the best instruction they had ever experienced. The course on networking was particularly effective. The material in the Theory of Operations course was excellent. Hewlett-Packard, though, should develop diskette versions of its on-line tutorials; we were only able to obtain CD-ROM materials.

Some of the most valuable training occurred outside the classroom. Hewlett-Packard's staff patiently explained how to implement components, configure systems, and tune applications. Again, the emphasis was on helping PCC become self-sufficient.

d. System Management:

Unix is not particularly suited to a commercial environment. It is difficult to handle security, job scheduling, spooling, backups, etc. Those who have relied upon MPE-XL, VMS, etc., might find great problems in HP-UX. Hewlett-Packard recognized PCC's problems and helped the college locate solutions from third parties. The college eventually acquired CA-UNICENTER from Computer Associates, a fine product that resolved many issues.

e. Software Issues:

PCC faced a significant number of compatibility problems with its third party software. We were concerned about whether various versions of HP-UX, Oracle, SCT Banner, Dynix, MicroFocus COBOL, etc., would operate satisfactorily on the HP 9000-890. In particular, HP-UX 9.0 was absolutely new; other companies did not have compatible products for such an operating system. There was a possibility that some of the vendors might point fingers at one another and deny responsibility for problems. It was agreed that Hewlett-Packard would provide specialists to work with third party software companies in order to resolve problems.

There were a few software issues that caught the implementation team by surprise. Some of the desired Hewlett-Packard system software (e.g., SwitchOver) would not be ready for our particular configuration for several months. One of the requirements in pursuing highly advanced systems is the need for patience; not everything can be delivered at once.

Hewlett-Packard has some excellent system software. Open Spool, Glance, OmniBack/Turbo, RX Forecast, PerfView, SoftBench, OpenView, etc. Customers should give serious consideration to those products.

f. Conversion:

One of the college's greatest challenges involved the conversion from the Honeywell-Bull to the HP 9000-890 platforms. Fortunately, the porting of software and data was relatively straightforward. It was a time consuming process, but not mysterious. Hewlett-Packard was quite helpful; we were loaned some software to solve some of the conversion problems.

Hewlett-Packard made a significant contribution to our network conversion. We were carefully guided in our effort to install LAN connections. The college installed TCP/IP links among the HP 9000-890 platforms, the Honeywell-Bull mainframe, and the wide area network. It would have taken several weeks for the college to perform such work on its own. Hewlett-Packard gave us excellent help; we were able to implement data communications systems that provided a graceful transition to our new environment.

PCC was astounded at the processing power of the HP 9000-890 platforms. Applications were ported to the one-way processor for testing before being put into production on the two-way processor. The difference in throughput was almost unbelievable. Most of the jobs that took an hour to run on the Honeywell-Bull took only two minutes on the one-way HP 9000-890. Performance can be significantly improved by keeping a careful eye on how the system is tuned.

g. Project Manager:

One of the historical complaints about Hewlett-Packard is that it is sometimes a confusing company; it is not always easy to do business. Placing an order for products, for example, is frequently a challenge.

Hewlett-Packard conquered many problems by creating an effective organizational structure. Hewlett-Packard assigned a project manager to PCC. It was his job to coordinate activities, answer questions, organize services, and limit bureaucracy. Much of the success of the implementation was the result of the Hewlett-Packard project manager; he cared about helping PCC achieve its goals.

The project manager was quite helpful in solving problems that the college had brought upon itself. For example, PCC forgot to acquire an uninterruptable power supply system (UPS). We didn't remember to include the UPS in our RFP specifications. The project manager helped us expedite the purchase of the necessary equipment even though it was not his responsibility.

The college tried to help Hewlett-Packard by developing its own effective organization. There were definite contacts for various aspects of the project -- system software, networking, operations, etc. Those who implement new systems must be cooperative with vendors; Hewlett-Packard cannot be fully effective if the customer creates a great deal of frustration.

New customers should consider having weekly or bi-weekly status meetings with their Hewlett-Packard representatives. Everyone should keep track of various projects and discussion items. The goal is to promote communication and a positive sense of direction.

h. Additional Servers:

Customers should discuss their strategies for the installation of additional servers. PCC was somewhat short-sighted in its RFP; we failed to specify an Internet firewall, menu server, electronic mail server for students, etc. Although we were committed to open systems and distributed computing, the college didn't understand the need for extra equipment.

Eventually, the college acquired several HP 9000 computers in order to have application-specific servers. It was unrealistic, given our strategic directions, to put all of our software on two Emerald machines. Although the two HP 9000-890s could certainly handle the processing load, we wanted additional servers to fulfill some of our design requirements.

One of PCC's goals was to help Hewlett-Packard do business with the college. As a result, in addition to issuing purchase orders for specific products, there were open blanket purchase orders for other items. So, when it became necessary to order more servers, printers, workstations, etc., the information services department could make a phone call and order equipment promptly.

i. Understanding Upgrade Paths:

Every computer company is challenged to develop new technology in order to remain competitive within the industry. Hewlett-Packard is no different; there is a firm commitment to research and product development. It seems that improved disc drives, microprocessors, printers, and LAN products are being announced on a daily basis. Customers should have discussions with Hewlett-Packard about their upgrade path. Making a non-disclosure factory visit to Cupertino, California should help customers understand their options.

j. Mileposts:

Major projects should be divided into manageable pieces. Those who install HP 9000-890 computers should talk to Hewlett-Packard about an implementation plan that will allow everyone to celebrate the accomplishment of certain goals. Otherwise, the project becomes a long ordeal with few rewards.

k. Architecture:

The HP 9000-890s are known for their substantial performance. Each machine has significant processing power, bus throughput, I/O performance, memory, etc. In major installations customers should review the need for at least one HP 9000-890 to be a "data warehouse." Applications can be installed on other servers that work cooperatively with such a machine. Such an architecture follows the successful model that was implemented at George Washington University.

l. HP 9000-890 Emerald SIG:

An HP 9000-890 Emerald Special Interest Group (SIG) is being formed. It is an opportunity for customers to share information and pursue common interests.

6. Summary:

The implementation of HP 9000-890 platforms has been a significant success at PCC. Some of the accomplishments involved the excellent technology that is inherent within the machines. They perform very well. More of the success, though, can be attributed to the partnership that was developed between the college and Hewlett-Packard. Technological issues are important but working relationships are critical.

Hewlett-Packard is a company that can be trusted. Customers need to build upon Hewlett-Packard's corporate strengths; there is almost no problem that cannot be solved if the customer creates the right opportunities. The Sandbox principle, which involves good communication, flexible negotiation, and positive decisions is helpful.

Every organization is different; there are always unique circumstances and strategic directions. Customers should take Hewlett-Packard into their confidence and find a way to be successful together. There should be a careful discussion about how each party can help the other achieve its goals.

Doing business with any computer company is like entering a marriage. There must be an effort to make each other happy and fulfilled. The alternative, a divorce, can be devastating for everyone. The best advice for customers and vendors is to find a way to make each other successful; there will be significant rewards for such effort.

7. Additional Information:

Those who would like additional information should obtain Hewlett-Packard publication #5091-7345E, which describes PCC's migration to open systems technology. Also, the April 26, 1993 issue of Open Systems Today has a two page article on the college's efforts.

PAPER NUMBER: 4033

TITLE: Client/Server for Neophytes

PRESENTER:

Bob Petrovic
Speedware Corp.
150 John Street, 10th FL
Toronto, ON M5V 3E3
CANADA
416-408-2872

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

PAPER # 4034

DOWNSIZING TO HEWLETT-PACKARD PLATFORMS

JOHN M. WOOLSONCROFT
CONCEPTS DYNAMIC, INC.
1821 WALDEN OFFICE SQUARE, SUITE 500
SCHAUMBURG, ILLINOIS 60173
708-397-4400

INTRODUCTION

The business environment of today -- fast-paced, competitive, customer-driven -- has forced companies to find ways to improve their decision making. In a marketplace that changes rapidly, companies that make quick, well-informed decisions become market leaders. On the other hand, companies unable to use corporate information to solve problems and meet market demands are finding it increasingly difficult to compete. Such companies will, over time, fall by the wayside in terms of market share.

For most corporations, the need to "open up" corporate information to analysis and decision making is running into a barrier created by their existing information systems. These existing "legacy" systems, built largely in the 1970s and early 1980s, were designed when corporate information was viewed as "data" (hence the term: "data processing"). In such systems, access to data is available to few people, usually on a batch-report basis. These systems, by their very design, do not offer the flexibility and access to information that today's rapidly-changing organizations demand.

To meet today's needs, a new definition, or model, for information systems must be introduced. That new model is Open Systems.

INFORMATION AS A COMPETITIVE TOOL

Business conditions today mandate that a company's information assets be used as management tools to help take advantage of market opportunities. Companies who want to remain competitive are seeking ways to implement flexible and easily managed computing environments that put accurate, up-to-date information into the hands of those who need it.

Forrester Research, Inc., a well-respected information technology research firm, has identified the growing trend in major corporations to include information systems as a competitive tool:

"The pressures driving FORTUNE 1,000 firms to look beyond the mainframe are not about to subside... the mainframe's cost and inflexibility will be highlighted as companies are forced to find new ways to compete.

Senior management, MIS, and end-users alike will be looking for alternatives. This will fuel the ongoing re-deployment of corporate applications to other platforms."

Wal-Mart and Toys-R-Us are two excellent examples of companies who decided, early on, to base their businesses on distributed, open systems that use information as a competitive tool. These two companies now dominate their markets and are causing their smaller, less responsive competitors to scale back or leave the business altogether. The competitive edge they have established is a direct result of their use of corporate accounting information to speed decision making about inventory levels, product pricing, etc. As a result, they have not only improved their service to customers, they have also dramatically reduced their operating costs below their competitors'.

Arthur D. Little's *Forecast on Information Technology* sums up the current views on information as a competitive tool:

"...though the integrated information system is not the only key to success for the enterprise of the mid-1990s, the lack of such a system will be one of the surest ways to guarantee failure."

DEALING WITH LEGACY SYSTEMS

Unfortunately, most organizations who would like to use information for competitive advantage are finding out that their closed, centralized computer systems make this nearly impossible. These systems, designed 10-20 years ago, cannot provide the access to information that today's business environment demands. As a result, people throughout the organization become disenchanted and dissatisfied with their company's information systems capabilities. This includes:

- The CEO who wants the flexibility to reorganize quickly and easily as market demands change.
- The CFO who wants to reduce the costs for computer hardware, software, maintenance and personnel and to optimize the company's long-term system investments.
- The CIO who wishes to increase productivity and have more choice of new, cost-effective software and hardware solutions.
- The programmers who want to spend their time creating new, valuable information systems rather than being tied up in program maintenance.
- And, most importantly, the users at all levels of the organization who want better and faster access to information.

As consultant Frank Gens stated in a recent interview with *Computerworld:*

"Corporate management won't be sympathetic to IS [not doing] something strategic because they were busy taking care of their legacy systems... The IS organization's role is to facilitate, not dictate, business change."

Strategic Maneuvers Are Inhibited

The rigid structure of legacy systems can limit a company's ability to adapt to changing business conditions. Often a company finds that system enhancements and modifications, needed to reflect new products and organizational structures, are expensive, slow and complicated to implement. Because mainframe applications are usually designed from the "top down," any change, no matter how minor, has an impact on the entire system.

Take, for example, the effect that an organization's restructuring has on its corporate accounting systems. If the hierarchy of an organization changes, even slightly, the entire system can be affected in ways that are often hard to predict. Expenses may be tied to the wrong cost centers, making it impossible to accurately assess business performance and profitability. This can lead to incorrect decision-making that adversely affects the company's performance. At its worst, this inflexibility prevents restructuring, even though it is needed to better respond to the needs of the business. In these situations, the company's information systems restrict corporate growth instead of supporting it.

Decision Making is Impeded

Making informed decisions is always critical to any company. But in the ultra-competitive business environment of the 1990s it has become the key to corporate survival. Companies that cannot track and quickly report information on revenues, expenses, cashflow, asset values, and other key areas are in particularly vulnerable positions. These financial numbers are to a corporation like vital signs are to an individual. If they are good, the person or the company will survive and thrive. If they are bad, immediate action must be taken to restore the person or company to health. A company that tries to run without rapid access to the proper information is in the same position as a person whose doctor is looking at last week's blood pressure. The information may still be accurate enough, but is the risk worth taking?

In the closed world of most legacy systems, information is not shared between systems and users have little real-time access. For example, the manufacturing system may not link cost information to the company's general ledger system. Data must be re-keyed many times from system to system because of this lack of integration -- a waste of time and resources. Decision making is adversely affected because the people who need the information cannot get to it in a timely manner. And when they finally do, it may contain errors because of the frequent rekeying. All of this leads to slow and inaccurate decisions - a deadly combination in today's world.

Maintenance Quagmire Delays Progress

Most legacy mainframe system users are caught in a terrible maintenance quagmire. In a traditional mainframe information systems group, so much time is spent on maintenance that little time is left for development of new, mission-critical systems. Studies have shown that as much as 80% of a typical information systems staff's efforts go into maintaining older systems. With numbers like these, it is easy to see why programmer productivity in delivering new applications is so low in many Information Systems (IS) departments.

One of the major reasons for this maintenance quagmire is that most existing mainframe software systems are full of what has come to be called "spaghetti" code. They are characterized by complex, layered architectures that have built up over years of development and modification. These characteristics create an environment in which simple fixes and enhancements take four times as long as the development of an entirely new application. Furthermore, with mainframe software systems there are very few development productivity tools available and, in addition, most of these older systems have very poor, if any, system documentation.

Because of the development backlog and the slow response to requests for changes, users become frustrated with a proprietary mainframe environment. As a result, user departments frequently create their own information solutions, further adding to the complexity of the overall computing system. As time goes by, the number and complexity of disparate systems grows and everyone sinks deeper into the quagmire.

Mainframe Costs Devour IS Budgets

In looking at the costs associated with a mainframe-based computer system, there are really two kinds to consider: direct and indirect. Direct costs include such items as salaries and benefits for support personnel, and acquisition and ongoing maintenance costs for hardware and software.

Studies have shown that, on average, a mainframe requires twice the staff, 500% higher maintenance costs, 250% higher license fees and 150% higher acquisition costs than a mainframe-alternative UNIX-based system like the HP 9000. In fact, the studies have shown that the costs involved in simply *maintaining* a mainframe system are higher than the total cost of acquiring a mainframe-alternative, UNIX-based system.

A good example of this is offered by Hyatt Hotels and Resorts, which shifted from a proprietary mainframe to an open UNIX computing environment. The initial hardware investment for Hyatt's UNIX system was significantly smaller than the costs required just to *support* their previous mainframe-based system. Hyatt says that they have reduced costs more than 25% with their new open systems-based solution.

In addition to high acquisition and operating costs, any upgrades to a proprietary mainframe system are usually quite expensive. Also, the customer is usually locked into a single vendor for obtaining these necessary upgrades. On top of all this, depending on configuration, mainframe systems are not as easily scalable as typical UNIX-based systems. This makes both functionality and power difficult to add as an organization grows.

Indirect costs of a computer system include such items as lost profits resulting from poor decision making due to a lack of access to information, as well as increased personnel costs for user departments forced to deal with systems that are difficult to use. In many organizations these indirect costs can be much higher than the direct costs of acquiring and operating the systems. For this reason, they must be factored into any analysis that looks at the true overall cost of information systems.

As you can see, the financial burden associated with legacy systems that use mainframe technology is forcing companies to explore new models for computing. According to a study done by Forrester Research in 1992, users have achieved an *"...average $1.7 million per year...in savings"* by pursuing mainframe-alternative computing solutions. These types of cost savings, which will be explored in a later section of this paper, are helping to drive the ROI numbers that clearly support the move to open systems.

OPEN SYSTEMS: PROVEN MAINFRAME ALTERNATIVES

Before going further, it may be helpful to take a moment to clarify what is meant by "open systems."

Open Systems Today, a leading computer industry publication offers this definition:

"Open systems integrate existing computer resources with the newest, most powerful, and cost-effective technologies in order to deliver the information users need to do their jobs.

Open systems require the purchase of standards-based, vendor-neutral information technology products of which UNIX is the most important example.

Open systems enable interoperability with, and portability to, hardware and operating systems of competing vendors."

Open systems break down the barriers that block the free flow of information within a corporation. They make use of standards-based building blocks to create an information system that adapts to changing business needs. With open systems, companies can easily install and integrate the latest and most powerful hardware and software tools.

UNIX International, the organization of nearly 300 companies who have joined together to assure the future of open computing based on UNIX, recently summed up the prospects for open systems in the decade ahead in this way:

"The most valuable type of operating system for the computing environment of the 1990s is, of course, open -- by which we mean that no one company or organization controls it; that it will

facilitate the building of interoperable solutions; that it will enable open network solutions; that it will incorporate the latest, most powerful technology; that it will run on every important computer platform; that it will provide ease of software portability; that it will adhere to industry standards, and that it will be made freely available to vendors, resellers, and users."

All of the factors mentioned earlier are leading companies to make the change from closed, inaccessible mainframe systems to more accessible "open systems."

In a June 4, 1990, *Computerworld* survey of 194 IS Chiefs, *"48% of the respondents said they were considering mainframe alternatives."* According to these IS executives, greater flexibility, reduced cost and greater user access were recognized as the primary benefits of moving applications off the mainframe. A 1992 Forrester survey showed additional evidence of the move to mainframe alternatives and stated that, based on the numbers, the move to open systems was rapidly accelerating. The Forrester study reported that

"...a staggering 80% of the 75 FORTUNE 1000 firms interviewed are looking to break the mainframe's grip on application processing."

Motorola's General Systems Sector is an example of an organization that has embarked on an ambitious open systems strategy. Corporate Vice President and Director of Information Technology, Bill Connor, has been quoted as saying, *"My feeling is that downsizing is a part of making American business competitive."* When asked which applications were staying on the mainframe in his sector of the company, Connor explained that the strategy is to *"...totally get off mainframes...There's nothing that should stay on the mainframe."* Quite a statement from one of the world's leading technology companies.

It is clear then that open systems provide a computing environment that supports the demanding business needs of the 1990s. Open systems give people throughout an organization better access to critical information. At the same time, open systems help reduce the cost of computing -- another mandate for businesses today.

OPEN SYSTEMS AND CORPORATE ACCOUNTING

Open systems provide a new model for computing that is changing the information technology landscape in many organizations. But given the wide range of applications that companies use in their daily business, which are companies targeting as the first to migrate to open systems? The accounting applications.

Accounting systems are at the very heart of every organization, so it is logical for companies to want to ensure that they are running efficiently and effectively. They provide critical, timely information that is used by everyone from senior management to departmental users to make decisions about the operations of the business. And because almost every other system in the company feeds into the accounting systems, they are the perfect choice as the first place to implement an open systems strategy. By choosing accounting software that utilizes the technologies of open systems, companies make it possible to share financial information seamlessly between applications.

Corporate accounting and financial departments at a number of companies are leading the move to open systems. According to *Computerworld's* 1993 forecast issue,

"financial departments will at least start to consider a full or partial move to open systems this year in an attempt to speed up day-to-day processes and make financial data more accessible... Because accounting, payroll, personnel and

applications of that ilk are more generic than, say, manufacturing systems, they're an easier target for ... a distributed computing model."

The growing trend toward open systems accounting is confirmed in studies done by several leading research organizations. For example, according to a 1992 *Datamation*/Cowen & Company survey, 7% of survey respondents were currently using UNIX-based accounting systems and an additional 6% were planning to move to UNIX-based accounting during 1993. Likewise, research done by International Data Corporation (IDC) shows that even at IBM mainframe sites, the move to UNIX-based accounting systems is gaining momentum. A recent IDC survey of 100 IBM mainframe customer companies showed that 7% say they have some portion of their accounting systems running on a UNIX platform, and an additional 21% expect to do so by 1994.

Often, the biggest impediment companies face in bringing the advantages of open systems to accounting is a sense of fear and doubt on the part of mainframe system users. These users have grown comfortable with the idea that only a mainframe-based accounting system can give them the full range of accounting functions, security and controls, not to mention transaction processing power, that their businesses need. And, up until recently, they were entirely correct.

Today, however, there are some new open systems-based accounting packages that demonstrate a thorough understanding of accounting issues *and* that are designed to take full advantage of the latest technologies. These new systems can easily match the capabilities offered by older mainframe systems.
They offer the functionality, security, auditability and controls of a mainframe, while providing equal or better performance, greater flexibility, and lower costs. Because of their central importance to future information systems plans, these new systems are quickly moving to the top of many companies' priority lists.

In addition to allowing users to access information from multiple systems, today's open systems-based accounting packages help users in other ways. The best designed applications are easy to learn, use, and modify. These advanced accounting packages offer ease-of-use features such as ring menus, zoom windows, operator prompts at each field, and arrow key or mouse "point and click" capabilities. As a result, users need very little training before they can work comfortably with a well-designed open systems-based accounting application.

LEVERAGING THE BEST NEW TECHNOLOGIES

It is impossible to talk about the strengths of an open systems-based approach to corporate accounting without discussing the underlying technologies that make open systems so effective.

Technologies such as the UNIX operating system, relational database management systems (RDBMS's), fourth-generation languages (4GLs), wide area networks (WANs), and executive information systems (EIS's) are key parts of any highly-effective mainframe-alternative accounting system.

These technologies, when used in combination, provide an accounting information tool that can support the needs of any size organization, from a small regional distributor to a FORTUNE-sized global corporation.

UNIX

"UNIX" and "open systems" are often used interchangeably to describe the new computing environment that we have discussed. This usage, however, is not entirely accurate. While most UNIX-based systems can truly be called "open systems," UNIX is not the only operating system that has some degree of "openness." UNIX is, however, the operating system that takes openness to its highest level.

According to Arno Penzias, Vice President of Research at AT&T Bell Laboratories, the original developers of the UNIX operating system,

"Strictly speaking, the term 'open systems' is tied to the notion that the supplier of the hardware and operating system need not be the same. The operating systems are specified by publicly-generated standards, not by manufacturers of hardware or by the operating systems themselves. UNIX... is the only operating system that is specified by a standard."

Thus, the main reason that UNIX is considered the most open operating system is that it is the most standards-based operating system available. Because a standard is by definition consistent, software and hardware that are UNIX-based can change, so long as the new pieces of the system are kept compatible with the standard. Standards allow for change and improvement while protecting information technology investments.

UNIX-based software that is developed to industry standards also has the advantage of being portable and scalable. Users are free to move from one hardware platform to another, so new and/or bigger machines can be implemented in the future. This means that the software system can easily change to meet a growing business' evolving needs. Being standards-based also means that applications are interoperable, or easily able to share information.

UNIX has established strong market presence as the operating system of choice in the area of open systems. This strategically positions UNIX for continued growth. UNIX is the only standard operating system that has been in use for twenty years and that will run on a full range of hardware platforms, including mainframes, minicomputers, servers, workstations and PCs.

Today, more than two hundred different computer vendors offer products based on the UNIX operating system. And there are over 25,000 commercial UNIX applications, ranging from small business management to large scale industrial applications, being used by companies in a wide variety of industries. Because of this widespread use, progress remains constant and enhancements to UNIX are continually released.

The future prospects for UNIX are quite strong, as confirmed in market research reports by several leading firms. For example, research from both IDC and InfoCorp shows that UNIX license sales will grow to $32 billion in 1995 from $18 billion in 1991. Dataquest Inc. predicts 1996 UNIX license sales will climb to $44.7 billion and that the compounded annual growth rate (CAGR) of UNIX units shipped from 1990 to 1995 will be a phenomenal 27.5%.

Thus, UNIX-based software represents an excellent choice for any company seeking mainframe-alternative solutions. This is true not only because of the open, standards-based technology, but also because the outlook for UNIX's continued success means that an increasing number of hardware and software solutions will be made available to users of UNIX-based systems.

Relational Database Management Systems

Up until the late 1980s, data management software had not kept up with advances in hardware. In recent years, new heights have been reached in terms of performance and data availability with the introduction of relational database management system (RDBMS) products, such as Informix's OnLine, that bring the power of mainframe computing to open systems platforms.

Open systems RDBMS's offer performance-enhancement features such as shared memory, a cost-based optimizer, and direct I/O. They also provide key distributed RDBMS features such as multi-threaded architecture, two-phase commit, declarative referential integrity, and stored procedures that allow information to be shared between multiple systems. Finally, to assure high availability of key information, these new open systems RDBMS's offer on-line archiving, disk mirroring, and fast recovery mechanisms.

Most importantly, relational database management systems offer users a fast, easy way to obtain information in a format that meets their needs. With an RDBMS, reports and queries can easily extract information from multiple database files simultaneously. This allows, for example, users of open-systems accounting solutions to execute queries from any field on any screen. As a result, they can easily review or examine all levels of detail about a given transaction without interrupting the flow of their work.

Advanced Programming Tools

No matter how good an application is when it is first implemented, business and organizational changes will force the modification of systems to meet new requirements. As we discussed earlier, in the traditional programming environment using third generation languages (3GLs) such as COBOL and BASIC, this ongoing program maintenance consumes large amounts of programmer time. In fact, up to 80% of a programming staff's time may get tied up in making these needed changes.

Fourth generation languages (4GLs) offer productivity improvement in the development of software code. These state-of-the-art tools require as little as one-tenth the lines of code needed by a 3GL to accomplish the same function. This means that fourth generation languages reduce initial application development time and significantly speed the process of modifying software. The result is that programmer productivity is improved tremendously with a 4GL, giving the organization more new applications and more application enhancements than it would otherwise get. This in turn results in significantly more satisfied users and managers throughout the organization, as well as more satisfied customers who come in contact with the company's information systems capabilities.

In addition to 4GLs, tools exist that can help software application developers by doing some of the routine programming steps for them. Computer-Assisted Software Engineering, or CASE, tools generate code automatically from a specification entered by the programmer. These CASE tools can add a great deal of value by freeing programmers from the mundane tasks involved in creating code.

You should keep in mind, however, that some code and report generators available in the marketplace promise more than they can actually deliver. These tools are helpful for creating very simple programs, but they do not produce or allow for sophisticated, highly-adaptable programs. The "pseudo-code" they use is limited, and the reports they produce may not suit the special needs of the company using them. Most programmers prefer to use a 4GL with its inherent flexibility, rather than dealing with the limitations and short-comings associated with many of the available code generation tools.

Sophisticated Networks

The movement to open systems hinges, in large part, on the ability to communicate critical information between disparate systems. Today's wide area networks (WANs) and local area networks (LANs) help achieve this interoperability of systems by allowing different sizes and types of computer systems to work together.

In an open systems environment, WANs and LANs do not require any particular vendor's computer system or proprietary standards in order to function. The UNIX operating system offers built-in networking capability and supports widely acknowledged standards, such as TCP/IP, OSI, NFS and DCE, for linking computers and creating distributed systems.

Today's ease of networking also allows for the scalability of hardware platforms to meet specific processing needs. For example, applications requiring high transaction volumes and/or large numbers of users can reside on large machines, while applications which have fewer users and/or transactions can be placed on smaller machines.

It is important to note, however, that interoperability goes beyond connecting, or networking, internal company systems. Interoperability often means support for links with systems outside the organization, through such technologies as electronic data interchange (EDI). Complex computing architectures, operating as a cohesive system both inside and outside the organization, are most capable of supporting a variety of business needs.

Executive Information Systems

We've already discussed how open systems help increase and speed access to information within an organization. But unless the users of that information are given the right tools for interpretation and analysis, not much is really gained.

According to the 1993 Forecast issue of *Computerworld,*

"There are signs that top corporate chiefs in marketing, sales, administration and other nontechnical areas are showing more interest in using technology to improve their own productivity..."

Fortunately, the latest software offerings in this arena are easy to use and provide the detailed analysis capabilities that corporate users need to make informed business decisions.

An important concept that has grown in use over the past several years is that of an executive information system (EIS). An EIS accesses data and presents it through spreadsheets, graphs, and charts for analysis. Users of a well-designed EIS do not need technical expertise or knowledge; they can easily and quickly produce visual representations and interpretations of accounting data to aid in their decision making. An executive information system is especially useful for managers involved in evaluating business performance and strategic alternatives based on company financial data.

The value of executive information systems to today's top decision makers was confirmed in a recent *Computerworld* and Andersen Consulting survey of top executives in 200 large U.S. corporations. The study showed that executive use of desktop technology is rising significantly. According to the study,

"Many (top executives) expressed interest in using knowledge-based and executive information systems."

PROCEED WITH CAUTION

When a company decides to make the move to open systems, it should do more than just "slap a new coat of paint" on an old system. In other words, rather than simply putting a new user interface on the system, or replacing one centralized system with a network of smaller systems, the company should focus on how to best support the current and future information needs of the business. Every aspect of the company's operations should be considered. Only then can the computing system be designed to support, rather than dictate, the company's operations.

Demand Full Accounting Functionality

In the specific case of accounting applications, it is usually best for a company to purchase new accounting software that takes full advantage of advanced technologies, rather than spending the time and resources to build the applications internally. Accounting applications by their nature are fairly standard, allowing good packaged solutions to meet the needs of a wide range of companies in a variety of industries. The key is to seek out functionally-sound and feature-rich accounting software with proven success in the marketplace. This way, companies can start with a solid foundation and can go on to tailor the software to meet their specific needs.

Buyers of accounting software should be especially cautious of software vendors who try to sell a system that offers "modifiability" as the product's strongest feature. Such vendors will try to sell a package that has only a shell of accounting functionality with the notion that the buyer can then use the advanced programming tools we spoke of earlier to modify the system to their needs. While advanced programming tools, like 4GLs, are very useful in *refining* software packages, companies should not be forced to use these tools to "reinvent the wheel" by writing most of an accounting system themselves. After all, one of the main reasons to buy an accounting package is to avoid the tremendous time and effort involved in developing a system from scratch.

The best way for a company to insure that the package will meet the functional requirements of their business is to:

1) involve users in reviewing the functionality of the alternative packages, usually through a detailed product demonstration
2) check the references of the possible software providers to determine whether they have proven results with their accounting software at companies whose needs are similar.

By choosing an accounting systems vendor that has proven knowledge of open-systems technologies and corporate accounting processes, buyers of accounting packages will increase the odds of success and, ultimately, the effectiveness of their accounting information systems.

One final note on purchasing accounting applications code. Most companies, as we have discussed, require the option of customizing application software to meet their specific needs. For this reason, it is important that application software licenses include full access to source code, preferably at no extra charge. This will give an organization the flexibility it needs to address changing business requirements.

Establish Consensus on Key Terms

Another potential problem to watch out for when selecting an open systems vendor is a lack of consensus as to what is actually meant by various technical terms. For instance, *"client/server"* can mean different things to different people, especially vendor salespeople. A software or hardware vendor may present an "open systems, client/server" strategy that is actually nothing more than a nice graphical user interface (GUI) to a mainframe-based application. In such a scenario, the cost and proprietary nature of the mainframe remains, and the added expense and complexity of maintaining PCs with additional interface software is introduced. It is important to gain up-front agreement on key terminology so that there are no surprises later on.

Avoid Old, Tired Technologies

Trying to bring mainframe-based applications into the world of open systems has other problems to consider.

Simply porting existing mainframe applications to a UNIX platform brings with it the burden of applications designed for the computing architecture of a decade ago. For that reason, buyers should avoid packages based on old architectures and tools. In the case, however, of a company with a large number of custom applications, porting may be an acceptable alternative to rewriting. The trick is not to bring all the software maintenance problems and inflexibility from the mainframe down to the new platform.

Don't Expect Too Much from PCs

At the other end of the spectrum from mainframe applications is the PC world. In some cases, downsizing may be defined as a *DOS-based LAN*, where processing and data are scattered across the desktops of the organization.

Unfortunately, local area netware, or PC LANs, do not come close to providing the security and controls that companies rely on in a centralized mainframe environment. In the worst case scenario, high-volume, mission-critical applications may not even execute in the PC LAN environment because of the lack of processing power.

PC LANs do allow users to share data and functions in a limited way. Access is open, but usually only one user interacts with one piece of data at a time. For example, in the typical world of PC software, one user might create a letter or spreadsheet and then release it for another user's review and revision. Such simple user interaction and utility sharing needs can easily be met by a PC LAN environment.

PC LANs, however, will not accommodate the transaction processing and data integrity needs of large corporations. In such organizations, especially in the financial area, the need for users to interact is much greater. Usually data is being entered to any given system by several concurrent users. Furthermore, data must frequently feed automatically, even instantaneously, into a related system. For example, billing entries may need to feed into the accounts receivable system. High-volume, on-line transaction processing can be better handled by a UNIX server environment, where data integrity tools are already in place and where processing power can be much greater.

The bottom line to consider when deciding whether to move applications off the mainframe to an open environment is that the design of the system, including functionality, should match the operational needs of the business. Open systems offer a tremendous amount of flexibility to an organization, but they must be based on a technological and functional foundation that supports the company's current business needs.

REAL-WORLD SUCCESS STORIES

To better understand what is involved in moving to open systems, and what benefits such a move can have, it may be useful to look at the experiences of some companies who have actually been through it.

What follows are three case studies of actual companies in different industries who decided to implement an open systems strategy. While their businesses are quite different, they all had very similar information systems requirements in order to run their companies more effectively. All three found many of the same benefits to open systems, including substantial reductions in costs.

Billion Dollar Food Processing Company

Profile:

A $1 billion division of a FORTUNE 200 company was time-sharing accounts receivable software from Computeristics on their corporation's IBM mainframe. Because their information processing was supplied by the parent company, the IS department for the division was small, employing only about 6 people.

A major problem faced by the division was timely access to their financial information. The division found batch input of transactions with overnight verification too slow -- errors were not dealt with until the next day. Responses to requests for information involved batch reports and took several days. Once the batch reports were prepared they had to be mailed or sent by overnight express to the division from corporate headquarters.

After considerable deliberation, the corporation decided that to manage the business more effectively, they had to bring their accounting operations into the 1990s. The plan that was developed involved moving off the mainframe entirely.

The first attempt they made at downsizing their accounting systems proved unsuccessful. It involved rewriting the COBOL-based mainframe accounting applications they had been using. Unfortunately, they found that in doing so they were perpetuating an outdated design based on past models of their business.

System strategy and requirements:

After their initial failure, the corporation set a bold new information-systems strategy. To increase their systems flexibility and reduce overall costs, the decision was made to go with a UNIX operating system, HP 9000 hardware and to use Informix's Relational Database Management System (RDBMS) and 4GL.

Among the goals that were set were to:

1) Integrate billing feeds at four locations with accounts receivable,
2) maintain mainframe-quality audit controls,
3) obtain, for the first time, real-time user interaction with on-line validation, and
4) set up on-demand reporting.

Solution milestones:

The project plan allocated seven months to installing and implementing a new UNIX-based accounting package. One month into the project the system was installed and testing of its functions began. As part of the testing process, users were heavily involved to make sure that the functions implemented would meet their needs, especially in terms of information access.

This user involvement proved to be an extremely important part of the project because as the users actually tried the system, and became aware of the possibilities for improvement, the "functionality floodgates" opened. Their wish list of customizations and enhancements expanded significantly as they saw what was actually possible given the capabilities of the new, 4GL-based system.

Over the following 6 months, because of the programmer productivity improvements resulting from the use of the 4GL, these user-driven improvements were easily implemented. By the end of the seventh month the mainframe was unplugged as planned.

Benefits:

Obviously the most significant impact of moving to a UNIX-based accounting system was the complete elimination of high mainframe-operations costs, as well as dedicated communication costs.

The change also meant that the division was able to take complete ownership and control of its own data, giving it better information access for decision making and greater accountability over its operations. In addition, because of the improved information flow, corporate A/R collection capabilities were streamlined and enhanced, resulting in better accuracy and reduced collection times - a major cashflow improvement.

Finally, users were delighted to have easy-to-use, on-demand decision support capabilities tied to the more timely and accurate data. In fact, the system proved so popular with users and IS alike that after its first year in operation it was voted the best application in the corporation by both groups.

Division of Thirty-two Billion Dollar Service Company

Profile:

The U.S. service division of a $32 billion, global corporation had 50 regional operating centers throughout the country and no internal IS staff. To run their accounting operations, the division was using outsourced MSA financials on a service bureau's IBM 4381 mainframe. Unfortunately, as a result of this arrangement the division found that it had no control over the rapidly escalating mainframe maintenance and upgrade costs that were being passed on to them by the service bureau. In addition, the division's users were extremely frustrated with very poor access to information and with the lack of service bureau responsiveness to requests for system modifications. On top of this, response to requests for new applications was also very slow.

System strategy and requirements:

Because of the cost-control and information-access problems associated with their outsourced systems, the company decided to terminate their outsourcing arrangement with the service bureau. The division management decided that all of their previously outsourced systems, including their accounting applications, were to be replaced with UNIX-based systems that offered greater functionality and ease-of-use. Their criteria for selecting an accounting package specified mainframe-quality auditability, security and controls. In addition, greater access to information for site managers and significantly reduced operating expenses were critical requirements.

Solution milestones:

The first step in the project was to pilot new operational software for job cost accounting and to hire and train a new IS staff capable of supporting the new systems. Training for these people included both UNIX and 4GL skills and required a total elapsed time of three months. After this training was completed, the rollout of the accounting applications to remote locations began.

Over a period of one year, the division's new financial accounting software was installed and implemented, including migration of existing financial data from the service bureau mainframe system. The company now operates an HP 9000 server at headquarters, and the division's financial transactions from across the U.S. are posted through a wide area network that connects all 50 remote locations to each other, as well as to headquarters. The company is pleased to point out that the project of migrating their financial applications to a UNIX environment was completed on time and within budget.

Benefits:

The new system is viewed as a complete success by company management, both functionally and financially. It met all of the users' expectations for increased functionality and enhanced information access, as well as their stated requirements for security, auditability and controls. The company's financial managers are pleased that the organization now has multi-location reporting capabilities with centralized cash management. A two-person IS staff now supports both the central computer and users at 50 remote locations. Thanks to the relational database and 4GL, these two people are able to maintain normal operations while still having time to create application enhancements and customized reports.

From a cost standpoint, overhead for such items as software license and maintenance fees has been substantially reduced. In addition, day-to-day accounting operations have been greatly simplified and redundant tasks have been eliminated. For example, the 17 individually-created spreadsheets previously used for period-end reporting are no longer needed. And because of improved information access, the user community within the division feels that it is empowered with information, not just data. In fact, each operating location has increased control over their computing tasks and needs, resulting in much greater satisfaction with the company's IS resources.

Two-Hundred Fifty Million Dollar Technology Company

Profile:

Another example of a company that successfully made the move to open systems is a $250 million high-technology company with 12 remote operating centers. The company found that its ability to make strategic moves in a highly dynamic market was hindered by the inflexibility of its computer systems. In particular, the rigid, layered architectures and 3GL code of its mainframe system were restricting the company's ability to modify its information systems to reflect the operational impact of organizational changes. The company was using MSA's financial accounting software running on an IBM 4381 mainframe. As a result, high operating, upgrade and maintenance costs for both hardware and software had become financially burdensome. In addition, a large IS staff was required to provide support, thereby increasing the costs associated with operating and maintaining the company's mainframe system. The company was also challenged by a growing number of incompatible user-created systems that were in widespread use throughout the organization.

System requirements:

The company decided that to continue to grow, it would have to combine its multiple, fragmented systems into a single, fully integrated, decentralized environment. To accomplish this, the company chose to move to UNIX-based open systems. They also decided that their new systems would be based on a relational database and 4GL tools to provide flexibility, easy access to corporate information, and a rich development environment that would allow for easy modification. The specific goals that the company set for their new systems were:

1) to fully integrate all systems in a decentralized environment,

2) to provide a rich development environment for internal IS use in creating and modifying company-specific applications,

3) to maintain the audit controls and accounting functionality of a mainframe system, and

4) to provide the flexibility necessary to accommodate the CEO's desire to be able to change the accounting systems to reflect the dynamic corporate structure.

Solution milestones:

The first step in the project was the complete retraining of the IS staff in UNIX and 4GL technologies. As this was being completed, the company installed new UNIX 4GL-based accounting software on a development platform so IS personnel could experiment without disturbing existing applications. In addition, the accounting software that the company selected was customized to their needs and installed for them.

After the installation of the new system, the data conversion process from the IBM 4381 was completed. Following this, the old and new accounting systems were run in parallel for a period of time during which the new system was fully tested. At the end of the test period the new system went live.

Benefits:

With the implementation of new UNIX-based systems, the company realized an 80% reduction in their overall information systems operating costs. A significant portion of that reduction came from the dollars saved due to reduced hardware and software maintenance costs. In addition, the new 4GL development environment led to a 500% productivity increase in application development. This has allowed new applications, needed to address dynamic business needs, to be completed in a fraction of the time they previously took.

The new UNIX-based systems have also resulted in substantial increases in the operational effectiveness of the company. The "openness" of the new systems means that all operational and accounting software systems are fully integrated and, as a result, company managers are better able to review and control financial activity and business performance at remote operating centers. As an added benefit, reporting-structure changes, needed as a result of the company's response to market opportunities, are easily handled with the more flexible open systems-based software.

OPEN SYSTEMS: DOLLARS & SENSE

It would be wrong to look at the many benefits of open systems without also focusing on the financial justification accompanying such a project. For top corporate decision makers, the bottom line on open systems is "Do they make dollars and sense?" As with all company investments, new computer systems must be measured by the return on investment that they bring to the corporation.

To give you some idea of the tremendous financial benefits possible with open systems-based accounting solutions, what follows are actual financial numbers from three major companies who made the move to open systems. Their returns on investment are typical of such projects and they show that UNIX-based corporate accounting solutions deliver tremendous savings over traditional mainframe-based systems.

Leading Transportation Company

A $300 million transportation company in the Midwest was paying $1.2 million annually in outsourced information-technology expenses before downsizing to UNIX systems. The company decided to move from outsourcing to an in-house UNIX-based system to trim costs. The total acquisition cost for their new system came to $670,000 for hardware and software. With the open systems UNIX-based solution, the company's annual information technology expenses were reduced to $500,000 - an annual savings of $700,000. As a result, their investment in the new system was recovered in just 12 months.

Major Retailer

A $1.5 billion retail company was paying $9 million annually in information systems costs before implementing an open systems solution. Their one-time acquisition cost for a UNIX-based system, including hardware, software and custom programming, totaled $4 million - less than half the yearly expenditure on their existing mainframe system. After downsizing, annual information systems costs were reduced to $4 million - an annual savings of $5 million. As a result, the new system investment was recovered within 10 months.

Rapidly Growing Technology Company

A $250 million health-care technology company was spending $460,000 annually on maintenance alone for their mainframe system's hardware and software. The company decided to downsize to a UNIX-based solution at a total cost of $650,000, including hardware and software. After downsizing, the company found that the annual maintenance on the UNIX hardware and software amounted to only $60,000 - an annual savings of $400,000. Thus, within 20 months the initial investment in the new system was recovered through savings on maintenance costs alone.

ISSUES TO CONSIDER WHEN MOVING TO OPEN SYSTEMS

The decision to move to open systems brings with it a wide range of challenges and opportunities for both the IS department and users alike. Careful planning and thoughtful consideration of the many issues involved can mean the difference between success and failure in implementing these new systems. The following are some key issues that have proven to be most critical in companies' open systems migration plans. While the list is not all-inclusive, it should help you in formulating a successful implementation plan that addresses the major IS and user department concerns.

Re-engineer business processes

Migrating to an open-systems environment provides an excellent opportunity to re-engineer existing business processes that have proven cumbersome and inflexible. Corporate users should ask themselves how many of the procedures they have in place have been imposed by the computer systems and applications they are using or have used in the past. Does the system adapt to and meet the specific needs of the company, or does the system force the company to adapt operations to it? Open systems offer companies the chance to re-design and simplify business processes to streamline operations.

Consider security and controls

Despite the impressions that people may have of UNIX from its early years, today's UNIX-based systems provide excellent controls and security features for corporate users. The best UNIX-based accounting applications are designed to mimic mainframe features in this regard. When considering the move to a UNIX-based accounting solution, it is critical that companies review the product to find out how the application handles transaction and data integrity. Any weaknesses that an accounting system has in this area can be easily and quickly revealed through careful questioning during software demonstrations.

Training is critical

Open systems require an entirely new skill set for IS professionals. They must not only understand broad new concepts such as open systems, distributed computing and client-server, but also develop skills on new products such as UNIX, TCP/IP, relational databases and 4GLs. Companies moving to open systems-based accounting systems should either re-train their existing IS staffs in these new skills or, if that is not possible, hire new programmers with hands-on experience in the technologies (UNIX, 4GL, etc.) in which the accounting software is developed. Project plans for implementing these new systems should always include action items to insure that IS personnel have the necessary skills to both install and support the new systems.

Communicate goals to all involved

The move to open systems almost always brings with it some rebellion among the information systems staff and often the system users as well. Many IS staff people may be concerned about job security while users may resist anything new and unfamiliar. The challenge for management is to address individual concerns while focusing on the benefits to the corporation.

One key to success is to insure there is a high level of communication with those staff members who will be affected by the new system. In the case of the IS staff, management should communicate its commitment to training existing staff in the new technologies. If staff reductions are necessary, management should let people know that the company will look to attrition, instead of layoffs, to achieve those numbers.

People should also be made aware that IS is implementing the new systems as part of a concerted effort to align IS goals with the organization's priorities. Goals for the project should be clearly defined and should be stated in business, not technical, terms, so the IS staff begins to focus on the welfare of the entire organization. Also, compensation and incentives should no longer be tied to the size of the department's budget or its headcount. Instead, salary levels and rewards should be based on the department's ability to solve the company's business problems with cost-effective solutions.

These ideas, coupled with examples from other companies' open-systems successes, should be broadly communicated to help motivate the IS staff. Likewise, users should be made aware of the many benefits that the new systems will bring. This includes making their jobs easier by allowing them more direct input to application design and development. By championing the new systems to IS staff and users alike, IS management can play a critical role in moving the organization to new levels of success.

Make it a team effort

Having the right partners can make all the difference to an open-systems implementation. Along the road to open systems, some difficult issues arise that can hamper the success of the migration process. One of these, for example, is data conversion. Getting data from the old proprietary systems to the new open systems can be one of the most difficult parts of the entire project. Without some help from outside experts, the project may slow down or even fail.

Fortunately, experienced professionals are available to handle these and other specialized tasks involved in moving to an open systems environment. An effective team can be assembled to address all hardware, software and integration concerns. This team should be headed by the IS department.

To be certain that a potential partner has the necessary experience, references should always be fully checked. Corporations need to make use of the expertise of those who have worked with open systems strategies, and who have proven results in making open systems work at a number of companies. If a potential vendor cannot supply real and recent references, the vendor should not be selected.

Open systems accounting is, as we discussed earlier, the heart of an organization's information systems. A company should not trust this valuable resource to anyone who does not have proven expertise in successfully implementing such systems for companies in a variety of industries.

CONCLUSION

Information management has become increasingly important to competitive success in today's business environment. Having the right information, easily available, at the right time can help an organization make strategic and operational decisions that lead to competitive advantage and corporate success.

Open systems offers companies the flexibility and ease-of-use they need to manage their information. Proven success stories from companies who have made the move to open systems show that for most large companies the operational benefits and significant cost savings of open systems far outweigh the potential short-term problems associated with moving to a new computing environment. And, when carefully thought through and planned, the transition to open systems can be handled very smoothly with both IS and user department personnel feeling good about the change to the new systems.

Accounting applications represent a perfect area for companies to focus their initial open systems efforts. Today's open systems-based accounting packages offer better performance, functionality, and user control than the mainframe accounting applications that they often replace.

In addition to saving companies money, these new systems leverage the technology strengths of UNIX, relational databases and 4GLs to deliver faster access to information, ease-of-use, ease-of-modification and greater interoperability with external systems.

For the IS department willing to accept the challenge, the move to open systems brings with it a redefined leadership role of even greater importance within the organization. An open systems-based computing strategy offers the IS department an opportunity to show a clear contribution to the success of the company, as well as significant cost reductions and improved ROI. Users also are quick to recognize the success of an open-systems implementation, and their overall opinion of the capabilities of the company's IS resources is usually dramatically improved.

The proven success stories from companies who have accepted the challenge of migrating to open systems demonstrate that open systems represent a new level of price/performance and flexibility for corporations. The demonstrated benefits of open systems are leading companies to view the open systems decision as one of "when?" rather than "if?". Companies who answer that question by developing carefully-considered, near-term plans for moving to open systems will be positioned to compete in the information-driven business environment of the 1990s.

BIOGRAPHICAL INFORMATION

John M. Woolsoncroft is the Director of Marketing for Concepts Dynamic, Inc. in Schaumburg, Illinois. In this role he is responsible for the planning and execution of marketing programs for Concepts Dynamic's open systems-based financial accounting software products.

Mr. Woolsoncroft has been involved in the selling and marketing of information systems hardware, software and services to major corporations for over thirteen years. Prior to joining Concepts Dynamic, he was with Andersen Consulting where he directed the worldwide marketing efforts related to Andersen's distribution and logistics software. Mr. Woolsoncroft began his career with IBM's Data Processing Division in 1980. He holds a B.S. degree in Business Administration from the University of Illinois and an MBA degree from the University of Chicago Graduate School of Business.

Mr. Woolsoncroft is a noted author and speaker on information technology and its application to business strategy. He has presented at the national meetings of such organizations as the Data Processing Management Association (DPMA) and the Association for Computer Operations Managers (AFCOM). In addition, his insights on trends in technology and marketing have appeared in such publications as Computerworld, Information Week, Sales and Marketing Management, and the Wall Street Journal.

SELECT BIBLIOGRAPHY

Bozman, Jean S. November, 1992. "Informix Chases DBMS Rivals." Computerworld.

Cunningham, Peter. September, 1992. "UNIX Means Business" supplement to Open Systems Today. pp. i-xii.

Faden, Mike. September, 1992. "When Downsizing Means Saving $60 Million for the Company." Open Systems Today. pp. 88-91.

Faden, Mike. September, 1992. "Can't Afford These Damn Big Glass Houses." Open Systems Today. p. 88.

Forrester Research, Inc. 1992. "The Computing Strategy Report: The Mainframe Voyage."

Gill, Philip J. January, 1992. "What is Open Systems?" UniForum Monthly. pp. 23-28.

Informix Software, Inc. 1992. "Informix and Hyatt Hotels Success Story."

O'Donnell, Patrick. June, 1991. "Don't Overlook Human Issues." UNIX Today! pp. 72-74.

Parker, John. Fall 1992. "Making the Unix Connection: Commercial Potential." VARBUSINESS. pp. 17-22.

Pike, Helen. November, 1992. "Unix's Corporate Climbing." InformationWeek. pp. 56-57.

Poole, Gary Andrew. January 1992. "Open Systems in 1992: How will the global market fare?" Unixworld. pp. 73-78.

Snell, Ned. November, 1992. "Mainframe Accounting Moves Down." Datamation. pp. 112-115.

Paper # 4035

HP Programmer's Toolset

New HP–UX Software Development Tools

Valerie Ho–Gibson
Computer Languages Operation
Hewlett–Packard Company
300 Apollo Drive
Chelmsford, MA 01824

1. Introduction

Software developers spend a large amount of time compiling, linking, debugging, and tuning their applications. These developers are continually looking for ways to improve their productivity. This paper describes the HP Programmer's Toolset, a new set of tools to assist the HP–UX software developer. The Toolset contains:

- The HP Distributed Debugging Environment (HP/DDE), providing increased debugger capability
- The HP Programmer's Analysis Kit (HP/PAK), providing enhanced performance analysis capability
- Blink Link, providing improved compile/link performance

This paper gives brief overviews of each tool, but focuses on HP/DDE, HP's strategic debugger. The section on HP/DDE begins with its functionality, then concentrates on the user interface and how it can be configured. Currently available on HP–UX Series 700/800 and Domain systems, HP/DDE will eventually replace the `xdb` debugger on HP–UX and MPE systems. There is also a brief discussion of some of the similarities and differences between HP/DDE and `xdb`.

The paper concludes with a description of how all the tools in the HP Programmer's Toolset can be used together effectively to improve programmer productivity.

2. HP/DDE – HP Distributed Debugging Environment

HP/DDE (henceforth referred to as DDE) is HP's strategic debugger. It is a powerful source–level debugger with a rich feature set. DDE supports the debugging of programs written in C, C++, FORTRAN, and Pascal. Highlights of DDE features include:

- Graphical user interface
  - multiple windows
  - user–defined menus, command buttons, and key bindings
  - context–sensitive pop–up menus
  - context–sensitive on–line help
  - easy customization
- Command language
  - macro definitions
  - conditional execution of commands (`if -then -else`)
  - execution of commands in a loop (`while -loop`)
  - language–sensitive expression evaluation
  - user–defined variables
- System support
  - support for shared libraries and dynamically loaded code
  - ability to attach to ongoing processes
  - ability to debug child processes

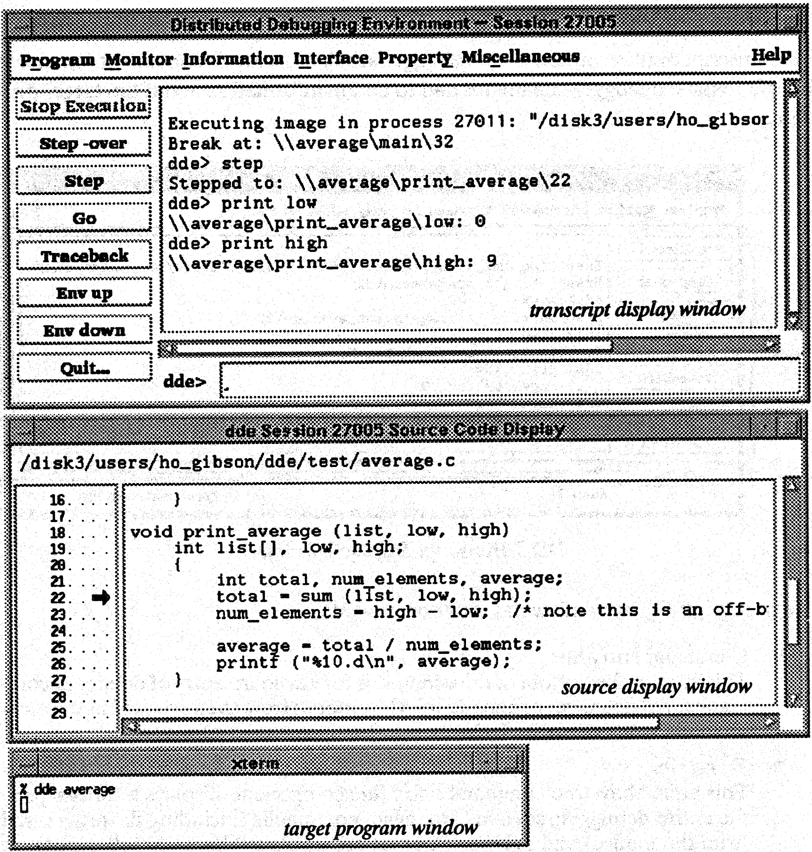
The remainder of this section focuses on the user interface. Command language and system support features are only briefly discussed in the context of debugger customization and HP's debugger strategy.

2.1. User Interface

DDE's user interface is based on the X Window System and OSF/Motif. There are up to six separate windows for displaying program information. Each DDE window has context–sensitive pop–up menus providing easy access to common debugger actions associated with each area of the display.

A non–graphical line–mode user interface is also provided. This user interface is particularly useful when an X environment is not available (for example, if you are debugging on a character–based terminal or remotely across a modem).

The following figure shows three windows in a typical DDE debugging session.


Distributed Debugging Environment — Session 27005
Program
Monitor
Information
Interface
Property
Miscellaneous
Help
Stop Execution
Step -over
Step
Go
Traceback
Env up
Env down
Quit...
dde>
transcript display window
dde Session 27005 Source Code Display
source display window
xterm
target program window

```
Executing image in process 27011: "/disk3/users/ho_gibsor
Break at: \\average\main\32
dde> step
Stepped to: \\average\print_average\22
dde> print low
\\average\print_average\low: 0
dde> print high
\\average\print_average\high: 9
```

/disk3/users/ho_gibson/dde/test/average.c

```
16. . . .      }
17. . . .
18. . . .  void print_average (list, low, high)
19. . . .      int list[], low, high;
20. . . .      {
21. . . .          int total, num_elements, average;
22. . ➡          total = sum (list, low, high);
23. . . .          num_elements = high - low;  /* note this is an off-b
24. . . .
25. . . .          average = total / num_elements;
26. . . .          printf ("%10.d\n", average);
27. . . .      }
28. . . .
29. . . .
```

```
% dde average
▯
```

Debugging Session with DDE

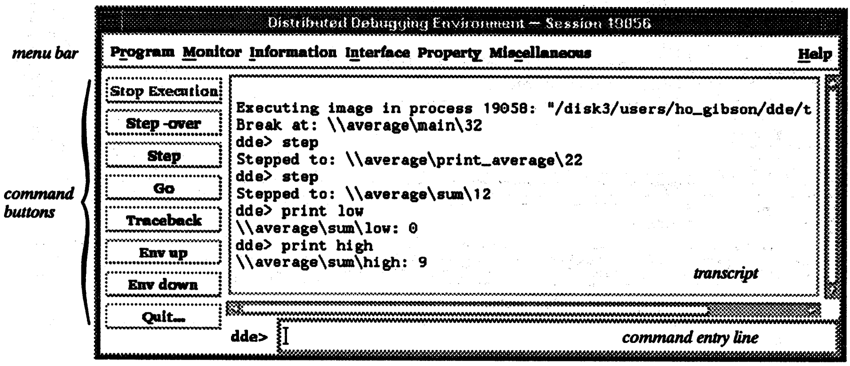
The following sections describe each DDE window in detail and also provide examples of how pop-up menus can be used to increase your productivity while debugging.

2.1.1. *Target Program Window*

The target program window is the window from which you invoke DDE. All input to and output from the program you are debugging is directed to this window. Separating program input and output from debugger input and output makes it easier to distinguish between the two and also allows DDE to debug graphics applications easily.

2.1.2. *Transcript Display Window*

The transcript display window is the debugger's main window. It contains components you use to issue debugger commands and to obtain information about the target program.


Distributed Debugging Environment — Session 19056
menu bar
Program
Monitor
Information
Interface
Property
Miscellaneous
Help
command buttons
Stop Execution
Step -over
Step
Go
Traceback
Env up
Env down
Quit...
transcript
dde>
command entry line

```
Executing image in process 19058: "/disk3/users/ho_gibson/dde/t
Break at: \\average\main\32
dde> step
Stepped to: \\average\print_average\22
dde> step
Stepped to: \\average\sum\12
dde> print low
\\average\sum\low: 0
dde> print high
\\average\sum\high: 9
```

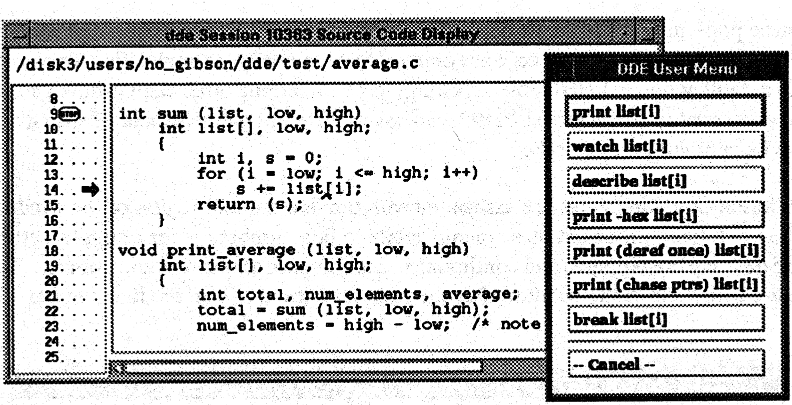
DDE Transcript Display Window

The transcript display window has four components:

- Command entry line
  This area, at the bottom of the window, is for keyboard entry of debugger commands. All debugger commands can be entered from the keyboard in the command line entry area, so DDE does not require use of the mouse.
- Transcript
  This area, above the command entry line component, displays a transcript of the entire debugging session. Debugger commands (including those accessed from the mouse) and debugger output are displayed here. Scroll bars allow you to scroll through your session.
- Menu bar
  The row of pull-down menus across the top of this display provides complete access to all debugger commands from the mouse. The commands are organized by function. For example, debugger commands associated with monitoring your program, such as setting breakpoints and watchpoints, are grouped under `Monitor`.
- Command buttons
  The column of buttons along the left side of the display provides quick access to common debugger commands. For example, the user can single step the program simply by clicking on the button labeled "`Step`". These buttons are user-configurable, so you can define your own buttons or replace the predefined ones with actions you commonly use while debugging.

2.1.3. *Source Display Window*

This window displays the source code for the program you are debugging.


dde Session 10383 Source Code Display
/disk3/users/ho_gibson/dde/test/average.c
DDE User Menu
print list[i]
watch list[i]
describe list[i]
print -hex list[i]
print (deref once) list[i]
print (chase ptrs) list[i]
break list[i]
-- Cancel --

```
 8. . . .
 9. . .    int sum (list, low, high)
10. . . .      int list[], low, high;
11. . . .      {
12. . . .          int i, s = 0;
13. . . .          for (i = low; i <= high; i++)
14. .                  s += list[i];
15. . . .          return (s);
16. . . .      }
17. . . .
18. . . .  void print_average (list, low, high)
19. . . .      int list[], low, high;
20. . . .      {
21. . . .          int total, num_elements, average;
22. . . .          total = sum (list, low, high);
23. . . .          num_elements = high - low;  /* note
24. . . .
25. . . .
```

DDE Source Display Window

The source code displayed is, by default, the source corresponding to the location where the target program will resume (or start) executing. The region on the left of the window displays file line numbers and possibly additional graphic symbols. For example, the arrow indicates the line of code where the target program is stopped. Stop signs indicate where breakpoints are set. Several other graphic symbols, not shown above, can also appear in this area.

2.1.4. *Context–sensitive Pop–up Menus*

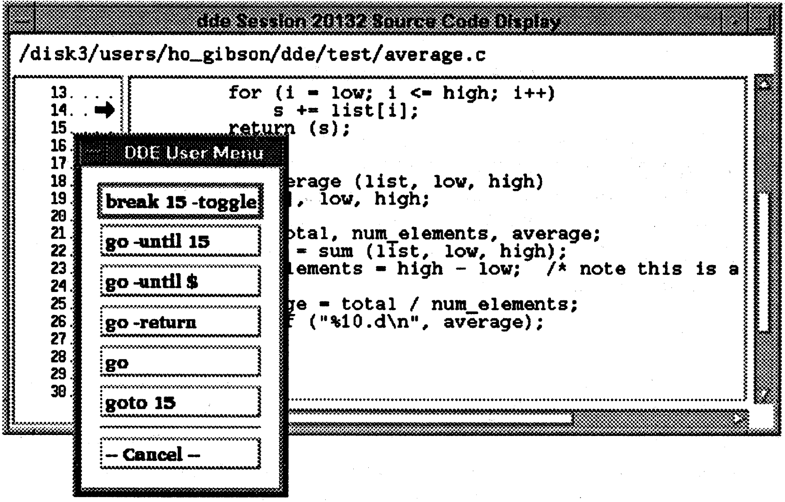
The context–sensitive pop–up menu is a powerful feature of DDE's user interface. It provides access to common debugger actions using the mouse. Clicking mouse buttons while the cursor is positioned in different areas of a window causes DDE to display different pop–up menus. For example, a single click with M1 (left mouse button) in the source code region brings up the menu seen in the figure above. The items in the menu are common actions you might perform while looking at the source code, such as printing the value of a variable or setting a watchpoint on it.

The text under the cursor when you click a mouse button determines the arguments to some of the DDE pop–up menu commands. In the source code region, DDE uses language–sensitive text selection to seed the contents of the pop–up menus. In the example above, the cursor was positioned at the opening bracket in the expression `list[i]` (note small ^ symbol on line 14). If the cursor had been positioned over the variable `list` in the expression `list[i]`, the menus would have been seeded with `list` rather than the entire expression. This capability makes it very easy to use the mouse to print any arbitrarily complex expression.

For even quicker access, double–clicking causes DDE to execute the first menu item instead of displaying the menu. In the example above, you can double–click on the variable in the source code region to print its value.

These pop-up menus can be defined by the user. You can easily reconfigure these menus to contain debugger actions common to your particular application, or change the default action (when double-clicking) to be something other than printing an expression's value. Each mouse button can have a different menu associated with it and can be configured separately.

Different pop-up menus are associated with the line number region of the window. The predefined actions in these menus relate to line numbers – for example, setting and deleting breakpoints, and continuing execution until a specified line number. The figure below shows the window after the user has clicked M1 on the line number 15.



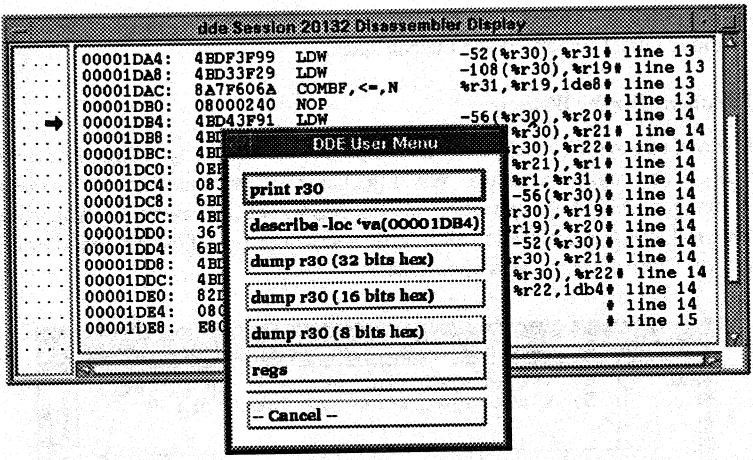
```
/disk3/users/ho_gibson/dde/test/average.c

13. . . .        for (i = low; i <= high; i++)
14. . ➡              s += list[i];
15.              return (s);
16.
17.
18.                        erage (list, low, high)
19.                        , low, high;
20.
21.                        otal, num_elements, average;
22.                        = sum (list, low, high);
23.                        lements = high - low;  /* note this is a
24.
25.                        ge = total / num_elements;
26.                        f ("%10.d\n", average);
27.
28.
29.
30.
```

DDE Source Display Window

As with the pop-up menus in the source code region, each mouse button has different pop-up menus associated with it. All the menus are user-configurable, and double-clicking always selects the default action (the first menu item) without displaying the menu.

2.1.5. *Assembly Display Window*

The assembly display window displays the disassembled machine instructions that correspond to the source code being displayed in the source display window. This window is not displayed by default. The figure below is an example of the assembly display window showing PA-RISC assembly code.


dde Session 20132 Disassembler Display
DDE User Menu
print r30
describe -loc 'va(00001DB4)
dump r30 (32 bits hex)
dump r30 (16 bits hex)
dump r30 (8 bits hex)
regs
-- Cancel --

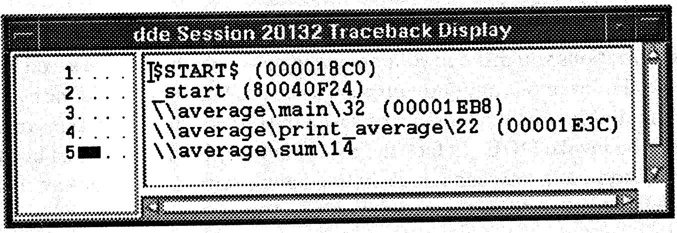
```
00001DA4:  4BDF3F99  LDW          -52(%r30),%r31# line 13
00001DA8:  4BD33F29  LDW          -108(%r30),%r19# line 13
00001DAC:  8A7F606A  COMBF,<=,N   %r31,%r19,1de8# line 13
00001DB0:  08000240  NOP                         # line 13
00001DB4:  4BD43F91  LDW          -56(%r30),%r20# line 14
00001DB8:  4BD                     %r30),%r21# line 14
00001DBC:  4BD                     r30),%r22# line 14
00001DC0:  0EB                     %r21),%r1# line 14
00001DC4:  083                     %r1,%r31 # line 14
00001DC8:  6BD                     -56(%r30)# line 14
00001DCC:  4BD                     r30),%r19# line 14
00001DD0:  367                     r19),%r20# line 14
00001DD4:  6BD                     -52(%r30)# line 14
00001DD8:  4BD                     r30),%r21# line 14
00001DDC:  4BD                     %r30),%r22# line 14
00001DE0:  82D                     %r22,1db4# line 14
00001DE4:  08C                              # line 14
00001DE8:  E8C                              # line 15
```

DDE Assembly Display Window

The pop-up menus in this display contain actions associated with assembly level debugging. For example, the pop-up menu shown above contains actions related to registers. Other menus in this display allow you to single step by instruction. As in the other displays, all these menus are also user-definable.

2.1.6. *Traceback Display Window*

The call/return stack appears in the traceback display window. The traceback display is automatically updated whenever your target program stops executing and DDE gains control. This window is not displayed by default.


dde Session 20132 Traceback Display

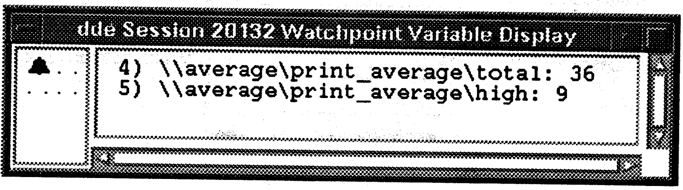
```
1. . . .  $START$ (000018C0)
2. . . .   start (80040F24)
3. . . .  \\average\main\32 (00001EB8)
4. . . .  \\average\print_average\22 (00001E3C)
5■. .     \\average\sum\14
```

DDE Traceback Display Window

Pop-up menus in this display allow you to change the environment of the debugger. For example, if you double-click over the name of the `print_average` procedure above, DDE changes the context of your environment to the point in procedure `print_average` where the procedure `sum` was called. The source display is updated to show the procedure `print_average`, allowing you to easily examine local variables in that procedure. Double-clicking in the traceback display window allows you to move quickly up and down the call stack.

2.1.7. *Variable Display Window*

DDE allows you to set a watchpoint on a variable or memory range to monitor changes in value as the program executes. When the value of a watched variable or memory range changes, DDE stops execution and notifies you of the new value. The variable display window is also updated to display the new value. By default, this window is displayed whenever there are watchpoints set.



```
4) \\average\print_average\total: 36
5) \\average\print_average\high: 9
```

DDE Variable Display Window

The figure above shows two watchpoints. The bell symbol indicates that the value of the variable `total` has just changed to 36. The variable `high` is also being watched, but its value has not changed.

DDE allows you to specify how often DDE checks watched values for changes (every instruction, statement, or routine). Since watchpoints cause your program to run significantly slower, the ability to control the level at which DDE checks for changed values can make a big difference in how quickly you can isolate your bug.

2.2. Customizing DDE

Any customizations you make to your debugging environment can be saved in a DDE startup file. For example, any pop-up menus and key bindings you define by using the DDE `menu` and `key` commands can be saved in your startup file so they are defined each time you invoke DDE. In fact, the menus and key bindings defined by DDE are predefined `menu` and `key` commands in the default startup file. You can modify the predefined menus by editing these commands and saving them in your own startup file.

In addition to creating or modifying predefined menus and key bindings in the user interface, DDE's command set also supports a powerful macro capability and commands to create conditional sequences or sequences associated with specific program events. For example, you could write your own DDE `alias` macro that uses DDE control commands such as `if -then -else` or `while -loop`. These features allow you to write sophisticated macros tailored for your specific program (such as a macro to print a complex data structure).

DDE also has predefined macros that you can use. For example, DDE defines the macro `'txt` to be the text under the cursor, and `'loc` to be the current line number. You can use these macros when defining your own aliases or modifying predefined menus.

All these features help you tailor DDE to your debugging needs, allowing you to debug your programs more quickly and efficiently.

2.3. HP Debugger Strategy

The HP debugger strategy is to support only one debugger, namely DDE.

Our strategy is to provide all essential `xdb` *functionality* (not exact *commands*) in DDE. Part of our current DDE development effort is to add this functionality to assist `xdb` users transitioning to DDE. Enhanced C++ support, localization, record and playback, and command–line editing are some of the features we are currently adding to DDE. When these features are available in DDE, we expect the transition for `xdb` users to be relatively painless. We also currently ship macros for common `dbx` and `xdb` commands.

Our strategy is to provide enhanced debugging support in DDE only. Future enhancements include support for debugging optimized code, multithreaded applications, distributed systems, and HP Micro Focus COBOL / HP–UX. We will continue to support `xdb`, but will not add any new enhancements. At a future date, `xdb` will no longer be available.

For readers familiar with `xdb`, the following sections provide a brief comparison of DDE and `xdb`.

2.3.1. Similarities between DDE and `xdb`

The DDE and `xdb` debuggers have more similarities than differences. Both debuggers provide users with the standard set of debugging functionality, including:

- program control (starting program execution, stepping by statement or instruction, delivering signals to a program)
- monitoring (setting breakpoints, tracing program execution)
- obtaining program information (displaying values of variables or memory locations, obtaining traceback of call/return stack)

In addition, both DDE and `xdb` support multiple languages (including language–specific expression evaluation), allow a user to simultaneously display source and assembly listings, allow a user to attach to a running program, and support debugging of shared libraries.

2.3.2. Strengths of DDE

There are several features in DDE that are not available in `xdb`. The most visible feature is the graphical user interface. As shown in preceding sections, DDE's interface is customizable, allowing you to tailor DDE to your own environment and making it easier to access the DDE commands you most commonly use. In addition to the user interface, DDE's powerful macro capability also adds to the flexibility of DDE.

A specific feature of DDE not in xdb is the ability to monitor a variable or an address range for changes in value (see *Variable Display Window* section above). While xdb assertions can be used to simulate DDE watchpoints, DDE watchpoints are easier to understand and to use. You can also specify the level of granularity of DDE watchpoints, whereas xdb assertions are evaluated after each machine instruction.

On Domain systems, DDE supports source–level debugging of optimized code and following child processes. These features are not available in xdb. We plan to make these features available in DDE on HP–UX and MPE systems in future releases.

In addition to these feature differences, DDE's internal architecture and design are superior to xdb's. The modular architecture of DDE enables us to better design and implement new debugger enhancements. The architecture of DDE was a primary factor in our decision to make DDE HP's strategic debugger.

2.3.3. Features of xdb not yet in DDE

The xdb debugger is smaller and faster than DDE.

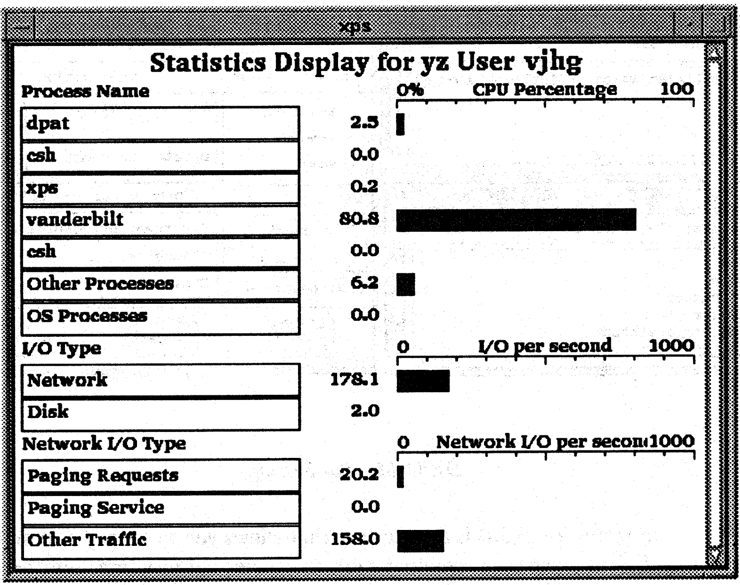
Other xdb strengths are its level of C++ support and localization. Additional xdb functionality not currently supported by DDE (Version 2.0) include command–line editing, record and playback, and integration with HP's SoftBench. Users who rely heavily on these features may want to wait for DDE 3.0 before transitioning to DDE. DDE 3.0 is currently targeted for release in mid–1994.

3. HP/PAK – HP Programmer's Analysis Kit

HP/PAK is a set of three program analysis tools. Each tool looks at performance from a different perspective: overall system resources, individual program performance, and program performance at the statement or instruction level. Each tool is described briefly below.

3.1. XPS

XPS provides an X–based view of process statistics running on a system. It enables you to see the relative use of system resources across all (or a subset of) the processes at the system level. Using XPS, you can determine if inadequate performance results from your program itself, or whether it arises because another process is running on the system concurrently. The following figure is an example of XPS output.



Statistics Display for yz User vjhg

Process Name	
dpat	2.5
csh	0.0
xps	0.2
vanderbilt	80.8
csh	0.0
Other Processes	6.2
OS Processes	0.0

I/O Type	
Network	178.1
Disk	2.0

Network I/O Type	
Paging Requests	20.2
Paging Service	0.0
Other Traffic	158.0

XPS Display

As XPS monitors your system, the barcharts are dynamically updated to reflect changes in relative usage on your system.

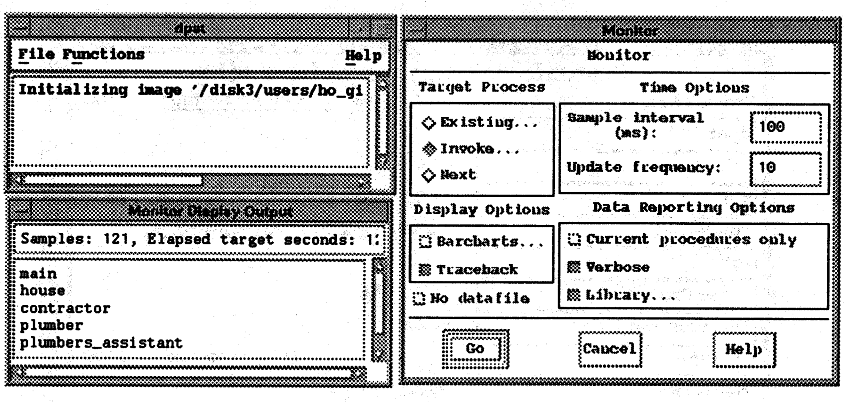
3.2. DPAT – Distributed Performance Analysis Tool

DPAT is an interactive tool that examines program performance at the procedure level. You can use DPAT to identify time–intensive procedures. By focusing your attention on those procedures that consume a lot of CPU time, you can maximize the impact of your performance improvements.

To use DPAT, you first use it to monitor your program, and then use it to analyze the results. You can use DPAT either to generate reports, or to interactively examine the collected data. Each of these functions is described below.

3.2.1. DPAT Monitor

DPAT collects statistics on your program during the monitoring phase. It does this by periodically stopping your program, obtaining the call stack at that point of execution, and then resuming your program. The following figure shows DPAT monitoring a program.



```
Initializing image '/disk3/users/ho_gi
```

```
Samples: 121, Elapsed target seconds: 1:
main
house
contractor
plumber
plumbers_assistant
```

○ Existing...
● Invoke...
○ Next

Sample interval (ms): 100
Update frequency: 10

☐ Barcharts...
☒ Traceback
☐ No datafile

☐ Current procedures only
☒ Verbose
☒ Library...

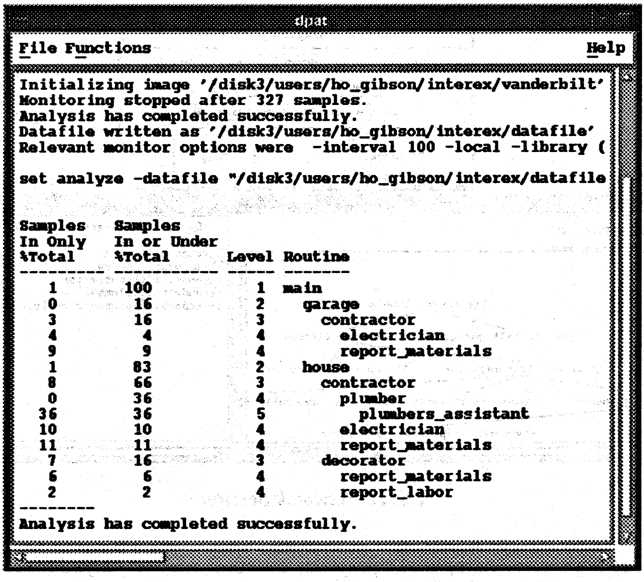
DPAT Monitor Session

The Monitor menu (on right) is an interface that allows you to specify parameters of the monitoring run, such as the command line to invoke your program, how frequently you want DPAT to sample the program, and what statistics you want DPAT to display while it is monitoring your program. In the example above, we configured DPAT to display dynamically the call stack of the program we are monitoring (seen in lower left–hand corner). This display is periodically updated to show the current call stack while your program is running.

3.2.2. DPAT Analysis

DPAT uses the samples (of the call stack) collected during the monitor phase to generate reports about the execution of your program. Through selections in the Analysis menu (not shown here, but similar to the Monitor menu), DPAT provides many different ways to analyze your data and to present the information.

The following example shows a hierarchical report. In this type of report, DPAT indents the procedures based on the call tree and shows how much time is spent in a procedure and how much time is spent in or under each procedure (in the procedure or any procedures called on its behalf). The reports can also be directed to a separate window or to a file.



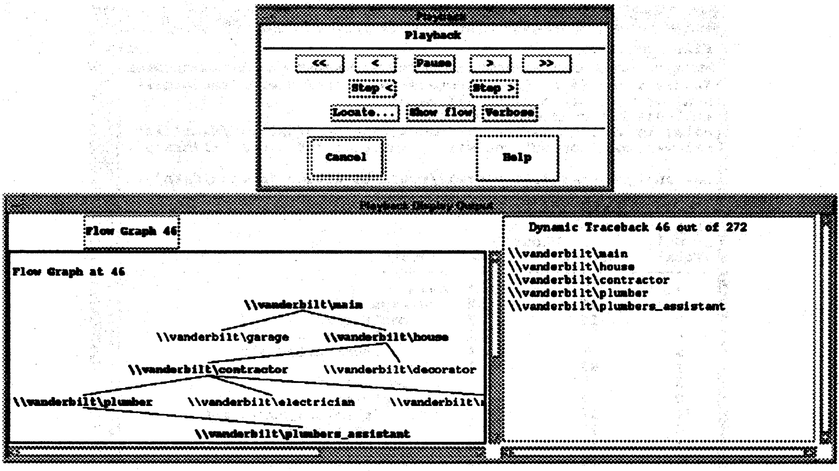
```
Initializing image '/disk3/users/ho_gibson/interex/vanderbilt'
Monitoring stopped after 327 samples.
Analysis has completed successfully.
Datafile written as '/disk3/users/ho_gibson/interex/datafile'
Relevant monitor options were  -interval 100 -local -library (

set analyze -datafile "/disk3/users/ho_gibson/interex/datafile


Samples   Samples
In Only   In or Under
%Total    %Total      Level Routine
--------- ----------- ----- -------
    1        100        1   main
    0         16        2     garage
    3         16        3       contractor
    4          4        4         electrician
    9          9        4         report_materials
    1         83        2     house
    8         66        3       contractor
    0         36        4         plumber
   36         36        5           plumbers_assistant
   10         10        4         electrician
   11         11        4         report_materials
    7         16        3       decorator
    6          6        4         report_materials
    2          2        4         report_labor
--------
Analysis has completed successfully.
```

DPAT Analysis Report

In the example above, you might want to start your performance analysis by looking at the procedure `plumbers_assistant`, since 36% of the execution time was spent in that procedure. By obtaining this information, you know where to direct your attention to obtain the biggest impact on performance.

3.2.3. DPAT Playback

Sometimes information on the behavior of your program can be gleaned by "watching" the call stack of your program dynamically. The Playback feature of DPAT provides a simple VCR–like interface that allows you to play through (at fast or slow speed) the samples DPAT has collected, single step frame–by–frame, or show the dynamic call tree of your program at any particular sample point.


Playback
<<
<
Pause
>
>>
Step <
Step >
Locate...
Show flow
Verbose
Cancel
Help
Flow Graph 46
Flow Graph at 46
\\vanderbilt\main
\\vanderbilt\garage
\\vanderbilt\house
\\vanderbilt\contractor
\\vanderbilt\decorator
\\vanderbilt\plumber
\\vanderbilt\electrician
\\vanderbilt\
\\vanderbilt\plumbers_assistant

```
Dynamic Traceback 46 out of 272
\\vanderbilt\main
\\vanderbilt\house
\\vanderbilt\contractor
\\vanderbilt\plumber
\\vanderbilt\plumbers_assistant
```

DPAT Playback Session

The figure above is an example of a playback session. We first played through the session and, in succession, DPAT displayed the traceback of each sample. We stopped playback by clicking on `Pause`, and have just selected the `Show flow` option. DPAT has displayed the dynamic call tree corresponding to the point at which we stopped playback.

By playing through the collected samples, you may notice that a procedure was called during a phase of your program when it shouldn't have been. DPAT's playback mode may help you discover algorithmic as well as performance problems in your program.

3.3. HPC – Histogram Program Counter

HPC analyzes compute–bound procedures at the statement or instruction level. Like DPAT, it monitors the execution of your program and then generates a report. The following example shows what HPC output looks like.

```
% hpc vanderbilt 10
Initializing binary vanderbilt
Monitoring stopped after 305 samples.
Total hits: 305    Total misses: 0
\\vanderbilt\report_materials
       68      31.1 *******************************
       69      32.1 ********************************
\\vanderbilt\electrician
       29      12.1 ************
\\vanderbilt\decorator
       80       8.9 ********
\\vanderbilt\contractor
       93       1.6 *
      103       4.3 ****
\\vanderbilt\plumber
       48       4.6 ****
\\vanderbilt\report_labor
       58       1.6 *
\\vanderbilt\house
      121       1.3 *
\\vanderbilt\plumbers_assistant
       38       1.3 *
\\vanderbilt\main
      156       1.0 *
```

HPC Output

In this example, HPC shows that most time is spent in the procedure `report_materials` on lines 68 and 69, so you may want to look there for possible performance improvements.

3.4. Using HP/PAK Tools Effectively

The three tools in HP/PAK can be used together effectively, each tool showing you a different view of your application's performance. For example, you may use XPS to look at overall system resource utilization by your application. You may notice an unusual amount of disk or network traffic when your application is running. Or, if your application has more than one process, you might use XPS to see how the processes compete for system resources.

If you identify a particular program that has inadequate performance, you can use DPAT to analyze its performance at the procedure level. Usually, you can make significant performance gains for your program by improving the performance of a few procedures, using DPAT to pinpoint these time–intensive procedures. If a more detailed analysis of a particular procedure or code range is required, HPC can be used.

4. Blink Link

Blink Link is an incremental linking facility that accelerates the link phase of the edit–compile–link cycle. Blink Link consists of a set of utilities compatible with *make*(1). To use Blink Link, you must first modify your makefiles to enable their use with Blink Link utilities. Once this is done, when your program is built, its object files are linked into a pre–link module (actually, a shared library) used to form the program. When files are subsequently modified during the development cycle, only recompiled object files are linked together to create a new version of the program. Because the pre–link module is not relinked, the link time for the recompiled object files is proportional only to the size of the recompiled object files, rather than the entire program. This results in faster link and build turnaround.

5. Using the HP Programmer's Toolset

The tools in the Programmer's Toolset can be used together in the following way to improve your productivity.

- Design your build processes to use Blink Link during the link phase.
- Use DDE to debug your program during development. As you make changes and rebuild your program, Blink Link will relink your modified object files quickly. Continue to use DDE and Blink Link iteratively until your program works correctly.
- Use HP/PAK to analyze the performance of your program, if necessary. Again, use of Blink Link will improve your build time. Continue to use HP/PAK, DDE, and Blink Link iteratively until the performance of your program is acceptable.

These basic development tools, provided in the HP Programmer's Toolset, allow all developers – novice as well as experienced – to be more productive during software development.

Paper Number 4036

Designing For Usability

Scott L. McGregor
President
Prescient Software
3439 Yuba Avenue
San Jose, CA 95117
Phone: (408) 985-1824
Fax: (408) 985-1936

ABSTRACT

There is more to designing a user interface than patching together various user interface elements. This paper postulates that the user interface should be designed to take advantage of the strengths of the "human machine" and avoid its weaknesses.

Scott McGregor has been developing and managing software development for over 20 years, including ten years at Hewlett-Packard, where, as manager of an R&D team managing software development of commercial CASE systems, McGregor developed the initial concept and prototypes for prescient agents described in this paper.

PAPER NUMBER:	4037
TITLE:	Performance and Capacity Management of Distributed UNIX Systems
PRESENTER:	Boris Geller BGS Systems 128 Technology Center Waltham, MA 02254 617-891-0000

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Systems Optimization Vs Migration to Open Systems

by
Elliott Chuang
EC & A
236 West Portal Ave
Suite No 238
San Francisco, CA 94127
(415)333-3666

Speaker's Bio:
Elliott Chuang is a 13 year information system technologist. He is a Certified Banyan Engineer, a Certified Banyan Instructor, a Certified Netware Engineer, a Lotus Notes Certified Specialist, a Microsoft Certified Professional for LANMAN, Windows, S2L Server, Mail and Workgroups. He has designed enterprise networks and managed large integration projects involving multi-vendor, LAN/WAN communications, Client/Server and object-oriented technology. He is a frequent industry spokesman and have appeared at the Downsizing Expo, OP/EN Expo, Interopt and Networld, Comnet and the Groupware 92. His writing has appeared in the Network News, and LAN Technologies.

Systems Optimization Vs Migration to Open Systems

Technological evolution and connectivity to the hp3000

Messaging and groupware as drivers of productivity

Online applications development utilizing 4GL and GUI

Servers as Open Systems platforms

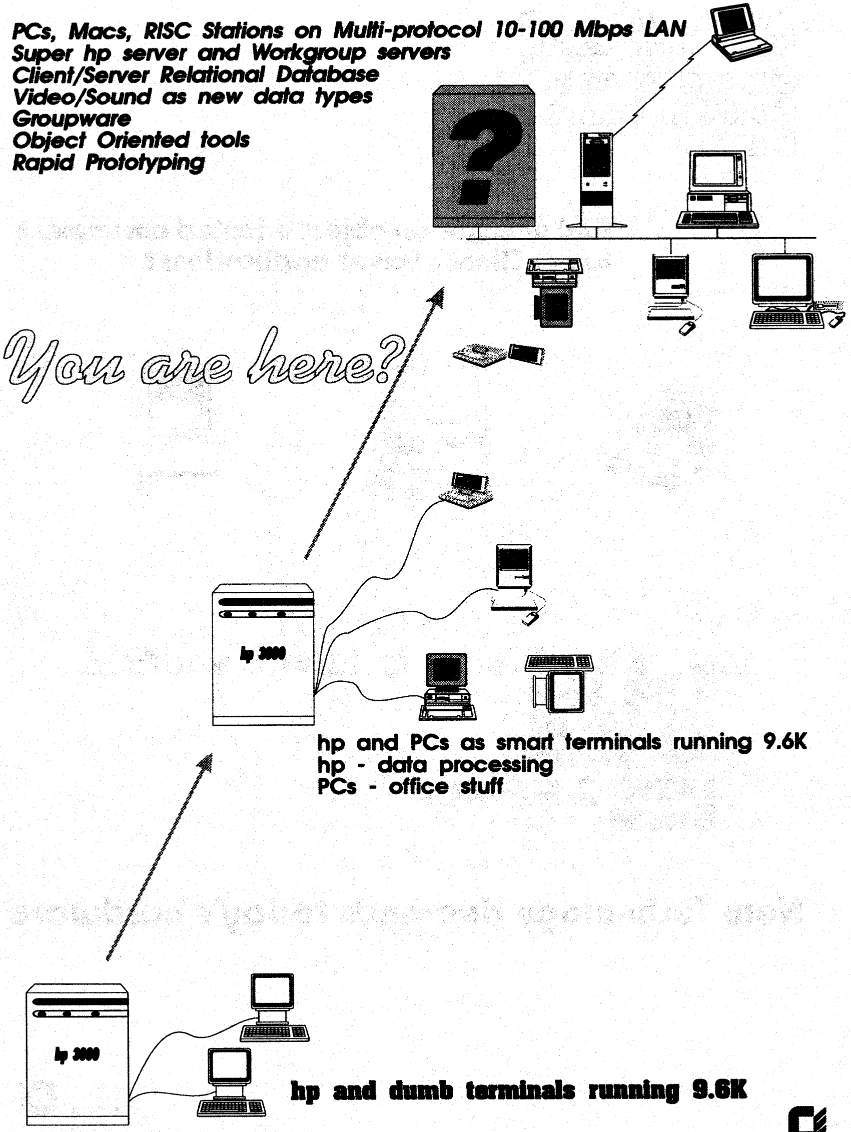
Relational versus Hierarchical database engines

Implementations issues:
Network platforms
Front end development tools

PC/LAN or UNIX

Migration rationalizations and realities

Migration strategies and pitfalls



PCs, Macs, RISC Stations on Multi-protocol 10-100 Mbps LAN
Super hp server and Workgroup servers
Client/Server Relational Database
Video/Sound as new data types
Groupware
Object Oriented tools
Rapid Prototyping

You are here?

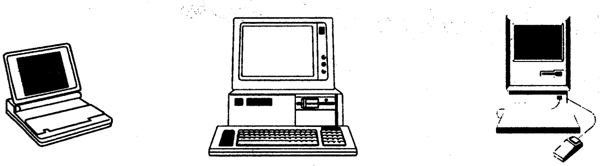
hp and PCs as smart terminals running 9.6K
hp - data processing
PCs - office stuff

hp and dumb terminals running 9.6K

PC were used for:

- Word Processing
- Shread Sheets
- Contact Manager
- Games

And now use an object oriented enviroment to run Client / Server applications?

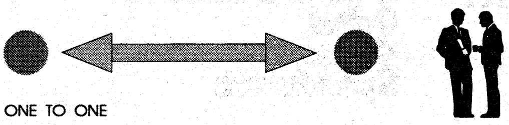


Must bring PCs up to todays standard:

- 8 MB RAM
- 100 MB HD
- Color/Video
- Pointing device
- Sound

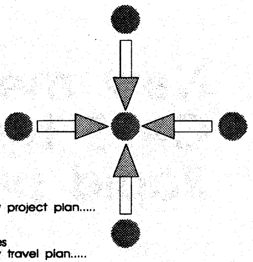
New Technology demands today's hardware

Messaging Systems - HP Mail
Scheduling / Calendar



ONE TO ONE

To: Jane.Mkt
From: Elliott.Corp
Please send me your market analysis......

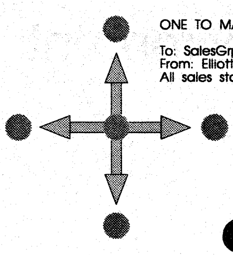


MANY TO ONE

To: Elliott.Corp
From: John.Eng
Attached is my project plan.....

To: Elliott.Corp
From: Mary.Sales
Attached is my travel plan.....

To: Elliott.Corp
From: Tom.Corp
Could you make friday's meeting.....

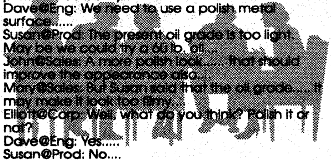


ONE TO MANY

To: SalesGrp
From: Elliott.Corp
All sales staff please attend new product training.....

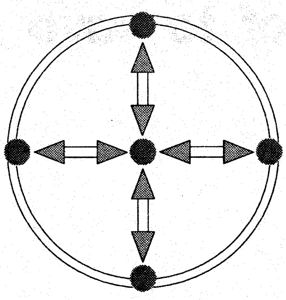
Groupware is:

Email, messaging, discussions, news, forms routing and checklist.



MANY TO MANY

Elliott@Corp: Is there a way to improve the products durability?
Dave@Eng: We need to use a polish metal surface......
Susan@Prod: The present oil grade is too light. May be we could try a 60 lb. oil....
John@Sales: A more polish look..... that should improve the appearance also....
Mary@Sales: But Susan said that the oil grade..... it may make it look too flimsy....
Elliott@Corp: Well, what do you think? Polish it or not?
Dave@Eng: Yes.....
Susan@Prod: No....
John@Sales: Yes....but....
Mary@Sales: Yes...John said it best...



On Line Applications

Database / Transactions Processing

Coming from:

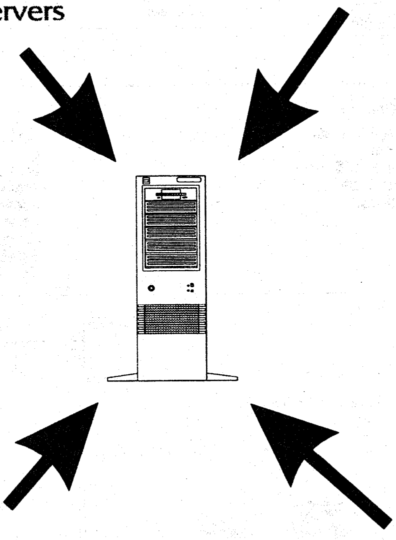
Image Database
Cobol
Cognos
Speedware

New methods:
CASE tools
Rapid prototype

Migrate to a GUI enviroment:
User demand
Better tools
Open systems

Not to save money $

Servers - Open O/S



U Brothers

Novell/USL/OSF
SVR4
SCO
various DB servers

Microsoft

NT Advance Server
SQL Server for NT

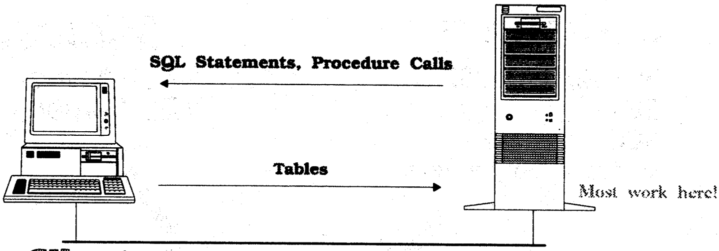
Novell

Netware 3.11
Netware 4.0
various SQL NLM

IBM

OS/2
CICS/OS/2
DB2/2
CICS/6000
DB2/6000

Relational SQL Engine



SQL Statements, Procedure Calls

Tables

Most work here!

Client

application program
screen forms
generation of SQL
application control
task switching

Server

optimize & execute SQL
manage transactions
business rule enforcement
stored procedures & triggers
security
concurrency management
logging & recovery
database creation & definition
data dictionary

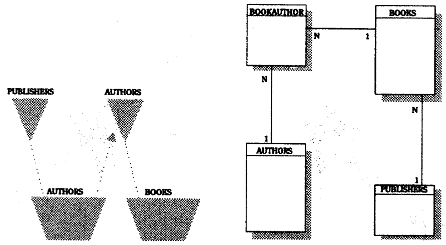

PUBLISHERS
AUTHORS
AUTHORS
BOOKS
BOOKAUTHOR
BOOKS
N
1
N
N
1
AUTHORS
1
PUBLISHERS

Image vs Relational

Client/Server DB Engines
hp Allbase SQL
Sybase/Microsoft SQL Server
Oracle
Informix

Implementation Issues

COMMUNICATIONS

Netware

IPX is easy on PC memory

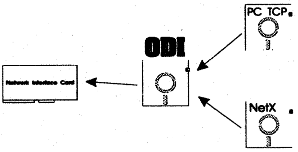
IPX supports PCs only and some flavors of UNIX (no Mac yet!)

IPX requires router or server present

TCP/IP on Netware clients requires ODI at the client

Multi-protocol stack requires a lot of memory

Difficult to have a single CONFIG.SYS that fits all machines



Software drivers does not 'take over' the network interface card.
Software drivers talk to NIC via the ODI Link Support.

Vines

Vines is big - it takes about 120K of client PC memory

Only a few high end routers can route Vines VIP

Requires PC based service
it should be set up as it own group to secure access

TCP/IP on Vines clients requires NDIS at the client

Multi-protocol stack requires a lot of memory

Difficult to have a single CONFIG.SYS that fits all machines

Implementation Issues

COMMUNICATIONS

TCP/IP

Industrial standard
but requires additional software on the PC
TCP/IP software can be expensive

Requires IP router to be present

Addresses have to be pre-defined

Multi-protocol stack requires a lot of memory

Difficult to have a single CONFIG.SYS that fits all machines

Netbios

Small packets - not good for lots of data

Generic peer to peer - requires no server or router present

No Network Layer (OSI) address

Must be bridged - not routable

Difficult to implement WAN

Implementation Issues

FRONT END - WINDOWS 4GLs

~Lots to learn - 600 Windows API calls + OO constructs.

~Pick Object Orient tools, as Windows predominates, OO will be widespread.

~Procedural programming falls apart for screen/event paradigm

~Development is tougher than you think

~GUI applications should be redesigned, not ported

~Language on the 3000 are less object oriented, requires retraining existing programmers

a few products:

Powerbuilder

SQL Windows

Visual Basic

FRONT END - Windows 4GLs

Powerbuilder SQL Windows

Development env. - Windows 3.1

Target env. - Windows 3.1, Mac in future

Target DB: hp Allbase SQL, MS/Sybase SQL Server, Oracle, Informix, DB2, and many others.

Generate 100% for target env.

Object oriented development

~ Transparent access to DB

~ Support for Windows OLE, DDE, ODBC

~ Suite of end user tools

~ Support OLTP standards

~ Version control

~ Stand-alone

~ One way export from CASE tools

~ Maintenance of object libraries

Pitfalls

PC LAN or UNIX?

PC/LANs ALONE are not YET the answer

Immature Operating System
~ Lack enterprise connectivity

Reliability

Global system management
~ PCs still need to be managed and configured

Complexity

But UNIX also means

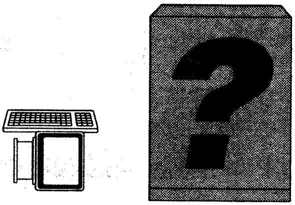
~few commerical packages
~limited system management
~high support cost
~complexity

Status of Client/Server

~Tools are not yet mature.
~Unexpected problems are likely:
performace problems, bugs, experiece.
~Lack of mini/mainframe fuctions:
Security, OLTP, data replication, etc.
~Cannot control bad run away queries.

No one vendor to call!

Reasons for migration



- ~Replace outdated systems
- ~Control costs
- ~Make effective use of PCs
- ~GUI
- ~User demands
- ~Connectivity
- ~Take advantage of Technology

Strategy

Use Client/Server tools for moderate-risk applications

- ~Migrate gradually
- ~Learn the new enviroment
- ~Keep number of vendors to a minimum

Use CASE tools for mission critical applications

- ~Rigorous analysis/design cycle
- ~Business re-engineering
- ~High cost, but appropriate for important applications

PAPER NUMBER: 4040

TITLE: Multimedia Support for Relational Database Management Systems

PRESENTER: Richard Kozicki
Empress Software, Inc.
50 Airport Parkway
San Jose, CA 95110
408-437-7724

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Interex '93
Paper Abstract

Paper #: 4041

Title: Testing System Performance by Emulating the Real World

Speaker: W. Duane Dorch
PERFORMIX, Inc.
3945 Freedom Circle, Suite 650
Santa Clara, CA 95054
phone: (408) 982-8989
fax: (408) 982-8988

Abstract: Performance is a critical factor for most multi-user systems. When you buy or develop new hardware or software, how can you be sure that the resulting system response time will meet your users' needs? Performance benchmarking using Remote Terminal Emulation (RTE) software reveals the virtues and inadequacies of a system before the system is exposed to the user community. RTE software stress tests a system and its applications by emulating the activity of actual users (unlike "canned" benchmarks) and gathering response time data. RTE software can be used to support computer purchase decisions, capacity planning, application and hardware development, quality assurance testing, and product marketing and demonstrations. With RTE testing, the key is testing in a realistic environment.

Please
Note: Handouts will be available at the time of presentation.

PAPER NUMBER: 4042

TITLE: Orientation to C++

PRESENTER:

Ray Swartz
Berkeley Decision/Systems
803 Pine Street
Santa Cruz, CA 95062
408-458-9708

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

ARE YOU MY PRIORITY?

JEFF ODOM

AUSTIN FOODS COMPANY
ONE QUALITY LANE
CARY, NC 27513
(919)677-3227

It is most probable that all of us have participated in a minimum of one time management course by now. It is also highly likely that we are using more than one of those modern "time saving" devices (computers, faxs, cellular phones, etc.). There is also a good chance we actually use our "time saving" devices to do more as opposed to saving time. We could actually say we probably are better educated in time use, do more in the time we have and ultimately, we remain a bundle of frayed nerves. Instead of having the four day work week promised a decade ago, we now are stressed out by the demands of intense multi-tasking.

Often we become unsettled if we do not have dozens of balls in the air simultaneously. We are stressed if we do, stressed if we don't. Where do we turn? Of course, we turn to our training! We change our "bad" behavior into "good" behavior. To do this we remember that they told us true change will not take place before three weeks but we will not change if we do not start within 24 hours of learning how. We must remember this is good for us (no pain, no gain?). Know your priorities, everyday study the list of time wasters, absolutely use the miracle tools the trainer so graciously let us in on, and on and on and . . .

In reviewing and participating in so called time management courses, I have found everything from outright product pitches to positive attitude promotions. One course I attended covered the following subjects all in a six hour ordeal:

- Personality profile
- Motivating others
- Concentration and alertness levels

- Task juggling
- Time management assessment
- Time management tools
- Delegation
- Procrastination
- Team building
- Positive attitude and words
- Lifestyle
- Job stress
- Assertiveness evaluation and training
- Communication
- Time waster tips

It seems the developer of the course needed some time management training to realize you can not adequately cover and integrate all of the above concepts in so short a time. Granted we need the whole of these concepts, if not a few more, to effectively manage our time but let us please have digestible portions! Further, when it is all said and done, how will I get done what I need done?

What is missing is the engine that drives the other time management techniques and tools. We need help setting priorities. What's important right now! I have even attended courses whose title included words such as managing or setting priorities, yet spent virtually no time on the subject. How could they with all the above listed topics to cover? What was their priority? Maybe it was as simple as duping us into one more time management course.

Actually, we are not putting down time management courses. Implementing the tools and techniques generally taught in them will clearly improve your plight. It was during a particular session whose title connotated help with priority setting but whose subject actually was a little time management tools and a lot of people management that my mind began to wander. How can we characterize and manage priorities? I began an animated daydream that paralleled P.D. Eastman's wonderful children's book Are You My Mother?. If you recall, the story is of a little bird who hatched while the mother was off looking for food. The hatchling wonders where it's mother is and begins searching, meeting other animals and some machinery in the search inquiring "are you my mother?" Ultimately the "Snort" (steam shovel) places the young bird back in the nest just in time for the mother's arrival with dinner.

If we could, let us take a few moments to share this adventure in recognizing the forces setting our priorities. But before we begin, take a moment to write on a piece of paper in priority order the 6 things you would do with your remaining time (including revenue generation) if you found out today that you would live in 100% health but die overnight exactly 6 months from now. What are the things you would want to do? Now set the paper aside and let us ask of the following tasks or personality traits found in others and yourself, are you my priority?

- ☺ The Bear - Threatening and powerful, successful at intimidating us.
- ☺ The Snake - Sneaky and cool, winding into our time.
- ☺ The Otter - Full of life and mischief, never a care for now, we will get to it later.
- ☺ The Lion - Loyal and powerful, seeing the world in the light of loyalty. Dignity and grace further adds to their influence.
- ☺ The Rhino - A very territorial beast who defends its space with charging and snorting. Not like a bully this one will defend to the death it's turf.
- ☺ The Kitten - So sweet and cute to "purrrrfection" ease you away from your sense of direction.
- ☺ The Rabbit - This one is cute but so jittery to boot. You begin to be attracted out of compassion for a creature such as this.

ARE YOU MY PRIORITY?

- ☺ The Elephant - The most commanding of all of our animals yet regal and unassuming, content with steering the course.
- ☺ The Steam Roller - Compacting and crushing all that falls in front of its path.
- ☺ The Fire Truck - Everything is a crisis, for that's it's job. Sometimes we wonder who's setting those fires?

Are these our priorities? Well yes in part, but the list you made a few moments ago are your true life priorities. All of the other things in your life must first be matched up to this list then set as a priority, based on your own time management technique (A, B, C's or I, II, III's, etc.). Use this list as your priority targets. From your target you can establish goals which will help identify your day-do-day and moment-to-moment priorities. Granted, the many influences of people and work tend to draw you this way and that, but keep it all in perspective of your larger whole life priorities. You will then stay on track with your goals.

Use the tools that work best for you in maintaining calendars, to do's, delegation and so forth. Remember the difference between important and urgent. Through people skills and assertiveness training, learn to manage the influence on your day-to-day priorities. Evaluate your priorities throughout the day, work in bite size pieces and avoid procrastination. These are all important to managing our time. The key though, is to thoughtfully evaluate what is important in the way you spend your time. Stop and evaluate "Are You My Priority?" based on a sound and healthy knowledge of what your true priorities in life are.

Volume Management

A Mainframe Implementation

By

Steve Cole

Northern Telecom

Paper No: 5001

Introduction

Volume Management or the use of *User Volumes* on the HP3000 systems is very misunderstood and greatly under-utilized. The benefits debated since the first MPE/XL systems shipped. This paper provides facts on user volumes permitting the individual system managers to make the appropriate decision concerning their site.

User volumes are not intended for everyone. A number of medium to larger sites observe significant benefits from implementing volume management while some smaller sites experience little or no benefit. Each site needs to make the decision to implement based on their needs and business requirements. Fears of a new approach or myths should not contribute to the decision.

User volumes on the MPE/XL systems are associated with the "*Private Volume*" on the older MPE/V platform by most users. Before continuing with the discussion I need to dispose of this misconception. The private volume implementation on MPE/V was added to the original operating system. The directory structure within MPE/V was not intended for private volumes and significant overhead was incurred when implemented, thus performance problems. With the MPE/XL operating system, user volumes are an integral part of the design minimizing overhead with its use. In fact performance gains from user volumes exceeds any overhead incurred with its implementation.

The physical commands required to implement are not covered in this paper. This paper discusses how user volumes integrates with the functions of the operating system to improve performance and provide improved system management capabilities. This paper provides an overview of the theory of system operations and how user volumes integrates with the system to provide improved performance.

Theory of System Operation

User Processing

The basic function of most computers today is to process transactions. The transactions are processed by user written programs inputs some raw data, processes this data to achieve some usable results and then outputs this data in a formatted form. To simplify further -- *Read, Process and Write.*

Read or Data Input Processing

The MPE/XL operating system determines the data needed before the program is given access to the CPU. The system checks if the required data is already in memory, if not the memory manager will fetch it. With the data in memory the program is launched and the processing begins. This method of operation assures the user the most effective use of the CPU once the program begins processing.

Data Output Processing and the Transaction Manager

Writes to MPE files, such as databases, native mode ksam files and log files on the MPE/XL systems are under the control of the system's *Transaction Manager (XM).* The transaction manager or XM is responsible for posting and ensuring the integrity of data on the system. When a program performs a write, the data is not physically written to the disk but instead handed off to the transaction manager. The transaction manager temporarily posts the data to the XM journal located in memory. The XM journal, containing transactions from a number of programs, is transferred to the XM log that resides on the master of the User Volume set. This process continues until a condition occurs that forces a *transaction manager checkpoint.* *(Several conditions that force a checkpoint are (1). a timer expires or (2). the XM log becomes half full.)* A checkpoint, occurring by user volume, is responsible for physically posting the transactions in the XM log to the user files. While a checkpoint is occurring in the background, transaction updates and other I/O is occurring on the user volume and its master.

It is not necessary that you understand the detail of the transaction manager. The points to understand are that data files are not directly updated by the program but by the transaction manager and that all transactions for a volume set are initially posted to the master of the user volume before the user files are updated.

Operating System Requirements on the System Disk

Along with the user program processing, the system itself requires a number of disk I/Os to operate. Unlike user data that can reside on any volume set, the system data or ***Transient Memory*** must reside on the MPEXL_SYSTEM_VOLUME_SET. The majority of the transient space resides on the master or the system drive. *(For the purpose of this paper I will refer to the system drive as Ldev 1 though no Ldev restriction exists.)* This high disk requirement on Ldev 1 for transient memory is because the system can be booted in single disk mode. Besides transient space, system spooling also requires space on the system drive and the system volume set.

System Spooling and Disk Fragmentation

Spooling not only increases the I/O on Ldev 1 but is a prime contributor to fragmentation on Ldev 1. One of the most frequent complaints expressed by HP3000 users is the amount of contiguous space required on Ldev 1 to perform an UPDATE. As no space recovery utility is available, the user is forced to perform an INSTALL. One of the key reasons for the fragmentation is spooling. Spoolfles are permanent files created in large number that have minimal life expectancy. As these files are created and deleted frequently, permanent holes are formed. By using *Volume Classes*, spoolfles can be isolated off Ldev 1.

System Directories

The directory structure on the MPE/XL systems is different from that on the MPE/V based systems. The MPE/XL environment uses a distributed directory system to keep track of users and files. This directory methodology spreads the I/O requirements across all the drives on the system but the main system directory containing the pointers to the other directories still resides on Ldev 1.

System I/O Requirements and I/O Bottlenecks

To summarize to this point, the master volume of a user volume set is where all transactions are logged to the XM log file before permanently posted to the disk. If your system has only the system volume set, the volume set master is Ldev 1. Along with the transaction manager functions, the system master is where the OS resides, the majority of the system's transient space, the primary system directory, as well as a place to keep spoolfles. By having only one user volume Ldev 1 has a tremendous I/O demand.

If we look at the disk hardware available today to accommodate this load we can expect only 27-33 I/Os per second for a disk drive. Consider that this load is placed on a single drive, the potential for I/O bottlenecks and reduce system performance is significant. I/O bottlenecks on Ldev 1 affects the performance of the entire system. Several sites that have upgraded CPUs found that their performance problem still remained. ***Figure 1*** demonstrates the demand that is placed on Ldev 1 when a single volume set is employed that must be satisfied with 27 I/Os per second.

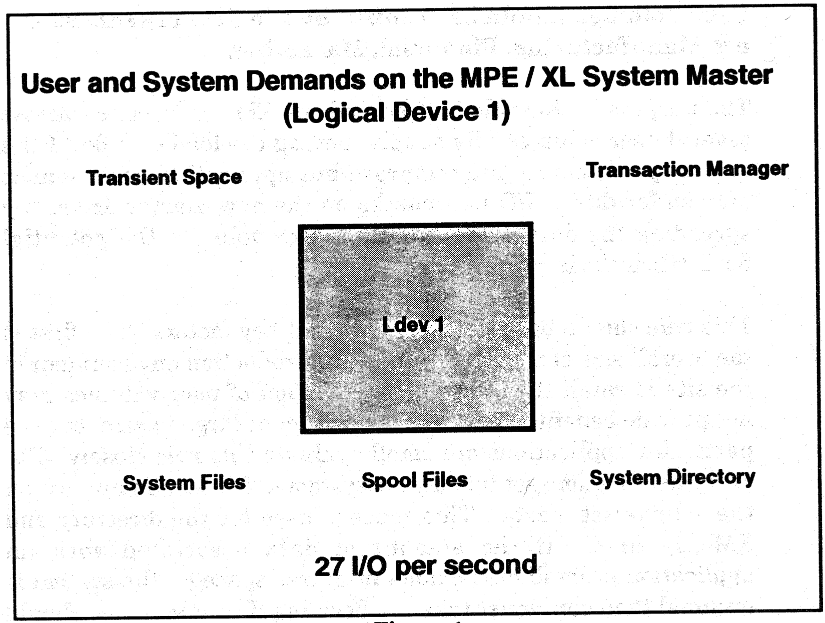

User and System Demands on the MPE / XL System Master
(Logical Device 1)
Transient Space
Transaction Manager
Ldev 1
System Files
Spool Files
System Directory
27 I/O per second

Figure 1.

Reducing the I/O Bottleneck

Implementation of user volumes can significantly reduce the I/O requirements on the system drive. The first key to implementing user volumes is to develop a strategy that you are going to operate by. Following are a few rules to assist in developing the strategy.

- **Remove all non-static files from the System Volume Set.**

  The significance of this rule is that it reduces the load of the transaction manager off Ldev 1. During check point processing the I/O rates on a system can increase 300 to 400 percent. By moving production MPE files, data bases, and KSAM files off the system volume set, the I/O and XM requirements impacting Ldev 1 are reduced.

- **User Volumes should be grouped by a logical organization, e.g. Manufacturing, Financial, Marketing.**

The purpose of this rule is to spread the XM transactions across several user volumes. By simply moving the load off Ldev 1 the system performance may improve but application performance may suffer due to I/O bottlenecks on the new master drive. By spreading the data across multiple user volumes the potential for bottlenecks is reduced.

This rule should be tempered by several key factors. The first is the overall size of the site. If the total production environment of the site is small then the implementation of user volumes may not provide benefit. If the site is medium or large in size, but the particular applications are small evaluate this rule closely. For every user volume set up on your system 400K sectors are lost on the volume set master. This space is used for the directory and XM log files. If the amount of data associated with an application is small or the amount of free space on the system is minimal then care must taken in deciding if user volumes should be implemented and how many user volumes should be created.

- **A User Volume should not exceed a size that can be recovered in a reasonable time.**

The philosophy that I have always expressed is that we design backup methodologies for system recoveries. When planning a user volume the time required to backup and recover should be the key driver. Whenever possible, the recovery time should never exceed the maximum time that an application can be down. A general rule that we try to follow is that a user volume never should exceed 20 gigabytes. Exceptions to this rule exist but it provides a target to shoot at. The average volume set size we strive to achieve is 10 gigabytes.

- **Data Base Log Files should reside on a different volume set than the Data Base.**

  This rule is valid only if *Roll Forward Recovery* is used to recover the Image data base. The is required as the transaction manager does not support dual commits. If *Roll Back Recovery* is used, the Image log file is required to be on the same volume set as the data base. *(By using Volume Classes within VOLUTIL the data base log files can be segregated to a single disk of the user volume.)*

  This rule is important for two reasons. The first reason is that it reduces the I/O requirements on the volume set master. The second reason is for recovery purposes. If a disk mechanism fails on the production volume set and a reload is required, the data base log file is protected on another volume set. Having the data base and its log file on the same volume set is counter productive. If a disk containing the data base is lost, the volume set requires recovery which includes the log file. If the user has one volume set, the system has to be re-installed before a recovery can be started.

Back Up and Recovery of User Volumes

The implementation of user volumes on your system increases the back up and recovery options available. Each user volume is a separate unit and can be backed up and recovered independently of the other. The exception is the MPEXL_SYSTEM_VOLUME_SET. If the system volume set is unavailable and requires recovery then an INSTALL is required. However, the data contained on the other user volumes are intact and a recovery is unnecessary. Once the system is installed and the system directory restored the access to the other user volumes and their files is re-established.

Following is some recommended rules that to consider when developing a backup strategy for your user volumes.

- **Backups of Systems using User Volumes should be performed by User Volume using the ;DIRECTORY option.**

  One of the key benefits of user volumes is the ability to back up and recover data by volume set. If user volumes are created to logically separate data, then backups can be performed in the same manner. By backing up the user volume and its directory, then recovery at a volume set level is easy.

  Encase of a disk failure requiring a reload of data, the user volume is scratched and initialized. Once initialized, the user volume is recovered by performing a RESTORE using the ;DIRECTORY option. Not only does this recover the data but also recreates the user volume's accounting structure. Only the affected user volume is unavailable and other users can continue to access the system during the recovery process.

- **The MPEXL_SYSTEM_VOLUME_SET backup should be performed using the ;DIRECTORY option.**

  One of the major concerns of most System Managers is how do I recover the system if a system disk fails. With the use of user volumes this concern is minimized. By developing a strategy that moves all user data off the system volume set then amount of data to recover is reduced. To recover, the system manager simply mounts the backup tape and start the install. If the system volume set is defined in SYSGEN, the system will build the system volume set automatically. Once complete, the RESTORE with the ;DIRECTORY option fully restores the system. By proper implementation of user volumes a large system can be INSTALLED and recovered in less than 3 hours.

Summary of Benefits associated with User Volumes

The discussion to now has covered the potential performance gains and the backup and recovery potential of user volumes. The paper at this point will focus on the benefits associated with a user volume implementation.

By removing all user data from the system volume set we can significantly reduce the transaction manager overhead from Ldev 1. This activity reduces the I/O load that translates directly into system performance. On some medium to large systems a performance gain of up to 30 percent can be obtained from this activity alone.

To reduce disk fragmentation on the system drive, add *Volume Class SPOOL* to at least one of the system drives other than Ldev 1. By using this volume class spoolfles can be keep off Ldev 1 reducing fragmentation associated with spooling.

The user data can be segregated by user class on their own user volume. This segregation can be by site or application class depending on what works best at your site. By setting up these logical units then backup and recoveries can be performed by the user class. Encase of a disk failure, only the user class on that user volume is affected. Along with the benefits obtained from backup and recovery, performance gains are achieved by spreading the transaction manager activity across multiple master drives. This increases transaction performance that translates to response time.

The overhead of user volumes is minimized by the distributed directory structure of the MPE/XL system. Directories are mapped into memory and pointers guide the system from one directory to another. As these directories exist on different drives whether user volumes are implemented or not, no performance impact occurs.

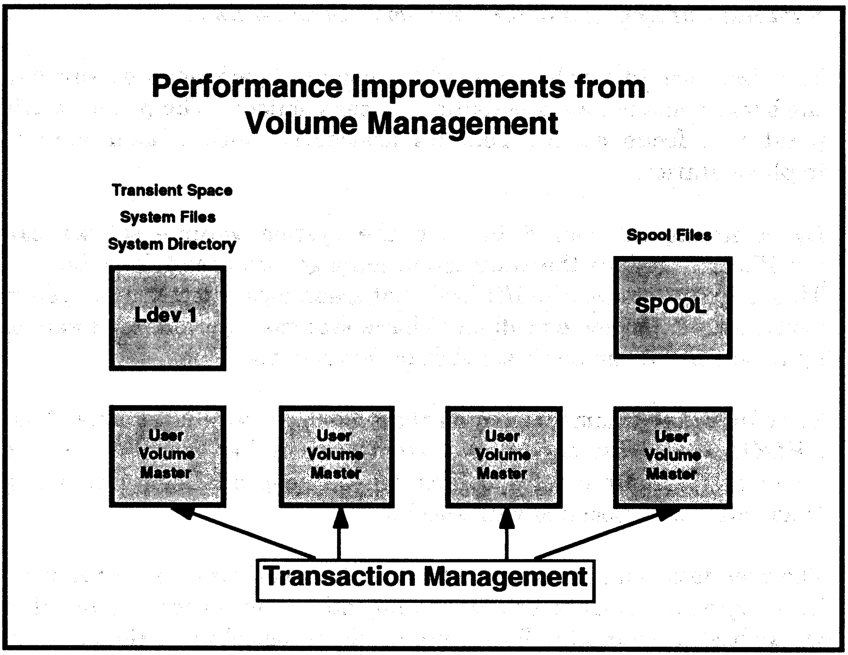

Performance Improvements from Volume Management
Transient Space
System Files
System Directory
Ldev 1
Spool Files
SPOOL
User Volume Master
User Volume Master
User Volume Master
User Volume Master
Transaction Management

Figure 2.

Conclusion

User volumes provides improved performance on the larger systems by primarily reducing I/O bottlenecks on Ldev 1. The performance gains achieved can be surprising depending on the transaction load. But performance is not the only benefit. Improved backup methods can improve your ability to recover and reduce down time.

The benefits of user volumes has been observed at a number of sites over the past several years. Some sites implemented user volumes to prevent problems while others try to reduce the impact if the problem occurs again. Some sites implemented user volumes to correct performance issues to save the expense of an upgrade while others tried to fix the performance problems the upgrade didn't correct. As we move into the day and age where uptime and performance is a requirement, can we afford to ignore any possible opportunities to do a better job.

Paper # 5002

MPE/iX Command File Tips and Techniques

by Jack Bailie

Exxon Company, USA
P O Box 2180, Room 2095
Houston, TX 77252-2180

(713) 656-2324

Introduction

Enhancements to the MPE Command Interpreter (CI) represent a major system improvement for MPE/iX over MPE V. The CI has become a programming language. Effectively using the CI in command files and UDCs requires a knowledge of the CI commands and skill in using them.

Several papers about MPE/iX command files have been presented at various Interex conferences and published in Interact. A list of some of these is at the end of this paper. These papers and Hewlett Packard Manuals present various aspects of using MPE/iX command files, and I highly recommend them.

Another way to learn about using commands is to peruse command files written by others. There are a number of command files on your system and in the Interex Contributed Software Library. You may want to examine the following files:

```
COMMAND.CSLXL
AUTOPAT.PATCHXL.TELESUP
TREELIST.PUBXL.TELESUP
MKACCT.PUB.SYS.
```

Even with all of this information available very little has been written about certain techniques, commands, conventions and standards. In this paper I will share with you, by example, some of the tips and techniques that I have adopted for command files. I will discuss style, techniques, parameters, predefined variables, help, functions, error handling, and debugging.

Style

There are few strict rules for command files. This is both good news and bad news. The good news is that you don't have to learn and follow a lot of syntax rules. The bad news is that it is possible to write command files that are unreadable and difficult to maintain. I have adopted my own guidelines for style. Let's look at an example - a simple command file to do a LISTFILE.

```
PARM fileset=@
PARM format=DISC

COMMENT  Title: LISTFILE a file set
COMMENT  File:  L.CMD
COMMENT  By:    J.D. Bailie
COMMENT  Rev:   3/12/91

LISTFILE !fileset, !format
```

Each parameter is on a separate line. Parameter names are meaningful. There are actually several good reasons for this that we will discuss later. Use upper case for keywords and commands and lower case for variables. Blank lines may be used to separate sections of code, in this case, the parameter list from the comments and the comments from the commands. Comments are used to document the file; the who, what, when information. I find that all of these conventions make command files easier to read and maintain.

Notice from the comment that this file is named L.CMD. It is a good practice to keep command files in their own group. I use CMD. The INTEREX CSL uses COMMAND. Either of these names or one of your own choosing is fine. I prefer the shorter group name.

Isolating command files in their own group prevents conflicts with other file names and makes them easier to maintain. Some system managers prefer to put their command file group in the SYS account. I don't put anything in the SYS account. I consider SYS to be HP's domain and I don't risk any chance of conflict.

You can give users access to your commands by setting the HPPATH system variable in a logon UDC to point to the command group.

The system default HPPATH is:

```
!!hpgroup,PUB,PUB.SYS
```

but you can change the HPPATH for example:

```
SETVAR HPPATH "CMD,PUB,!!hpgroup,PUB.SYS"
```

This command sets the path to the account command group, CMD, the account PUB group, the users current group, and PUB.SYS.

Parameters

The example command file above may be used as shown in the following examples:

```
:L  C@
:L  C@, 3
:L  FILESET=C@
:L  FORMAT=DETAIL,  FILESET=@
```

The first two examples use positional parameters, that is, the parameters are determined by their positions. The fileset is C@ and the format is blank or 3. The other two examples use keyword parameters. The parameters are determined by the keywords "FILESET=" and "FORMAT=". Note in the last example that the parameters are passed in the reverse order from the way they are defined in the command file.

You probably wouldn't use this technique for this particular command file, it is only used to point out the technique. If a command file has several parameters, with many that are optional, keyword parameters allow you to specify individual parameters without having to remember the parameter positions or count commas. Using meaningful parameter names, as suggested earlier, makes using keyword parameters easier.

ANYPARM

I have searched the LASER ROM CD and have concluded that, for some unknown reason, Hewlett Packard has not documented the ANYPARM parameter. ANYPARM may be used instead of PARM as the last parameter of a command file. In the example file above we could substitute:

```
ANYPARM format=DISC
```

ANYPARM captures everything that is entered beyond the last parameter separator, including any punctuation and spaces. Just as a regular parameter it may be assigned a default value. By substituting the ANYPARM parameter specification in our example, we can now execute the command as:

```
:L C@, DETAIL; PASS
```

Now the format argument becomes:

```
DETAIL; PASS
```

and the LISTFILE command becomes:

```
LISTFILE C@, DETAIL; PASS
```

If the user has sufficient capability, the PASS parameter will cause the command to display file lockwords and creators.

HELP

The MPE HELP command may be used with command files. For our example command file the MPE HELP would be:

```
USER DEFINED COMMAND FILE:  L.CMD.CAPS

PARM fileset=@
ANYPARM format=DISC
COMMENT  Title: LISTFILE a file set
COMMENT  File:  L.CMD
COMMENT  By:    J.D. Bailie
COMMENT  Rev:   3/12/91
LISTFILE !fileset, !format
```

MPE HELP shows that it is a user defined command file, the name of the file and the entire contents of the file (except for the blank lines that we added for readability). For complex command files this can be confusing and very little help to users that do not understand command programming.

You can make MPE HELP a little more useful by adding comments to the command file that describe the use of the command. For instance:

```
COMMENT
COMMENT  L - LISTFILE a fileset
COMMENT
COMMENT  Syntax
COMMENT   :L fileset [,format]
COMMENT
COMMENT  Parameters
COMMENT   fileset  - The file[set] to LISTF; default @
COMMENT   format   - The LISTFILE format; default DISC
COMMENT
```

Using this technique MPE HELP will list the comments, giving the user more information. The disadvantage is that it still lists the entire contents of the command file which may confuse the user.

A better technique is to use ECHO commands to provide the command help. For example:

```
PARM fileset=?
ANYPARM format=DISC

IF "!fileset" = "?" THEN
  ECHO
  ECHO L - LISTFILE a fileset
  ECHO
  ECHO Syntax
  ECHO  :L fileset [,format]
  ECHO
  ECHO Parameters
  ECHO  fileset  - The file[set] to LISTF; default @
  ECHO  format   - The LISTFILE format; default DISC
  ECHO
ELSE
  LISTFILE !fileset, !format
ENDIF
```

Notice that the first parameter of the command has been given a default value of "?". If the command is used without any parameters or if it is used with a question mark, the ECHO commands show the user the description of the command.

The output would look like:

```
L - LISTFILE a fileset

Syntax
 :L fileset [,format]

Parameters
 fileset  - The file[set] to LISTF; default @
 format   - The LISTFILE format; default DISC
```

This technique is much clearer and provides only the information that the user needs.

It you want to be fancier you can use line drawing characters so that this:

```
IF "!fileset" = "?" THEN
  SETVAR S CHR(14)
  SETVAR B CHR(14)+':'+CHR(15)
  ECHO
  ECHO ![S+'R,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,T']
  ECHO ![B+'   L - LISTFILE a fileset                         '+B]
  ECHO ![S+'@999999999999999999999999999999999999999999999999[']
  ECHO ![B+'   Syntax                                         '+B]
  ECHO ![B+'     :L fileset [,format]                         '+B]
  ECHO ![B+'                                                  '+B]
  ECHO ![B+'   Parameters                                     '+B]
  ECHO ![B+'      fileset  - The file[set] name to LISTF, @   '+B]
  ECHO ![B+'      format   - The LISTFILE format; DISC        '+B]
  ECHO ![S+'F,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,G']
  DELETEVAR S,B
ELSE
  LISTFILE !fileset,!format
ENDIF
```

Would result in this output:

```
L   LISTFILE a fileset

Syntax
:L fileset [,format]

Parameters
fileset   - The file[set] name to LISTF; @
format    - The LISTFILE format; DISC
```

In the examples that follow I will leave out the help code to shorten the examples. However, I recommend that you always include help, except in the very simplest command files.

Variables

Let's shift gears now and look at variables. We'll look at an example named FIND. We are going to build on this example through the next several topics.

```
PARM file

LISTFILE !file, QUALIFY
SETVAR _find_creator FINFO('!file','CREATOR')
ECHO Creator is  !_find_creator

DELETEVAR _find_@
```

This example uses the SETVAR command to set the value of a variable. In this case, since the name of the command file is FIND, I use a variable name that begins with "_find_". This naming convention assures that variables in the command file do not conflict with variable names in other command files called by your command file. It also makes it very easy to clean up after yourself at the end of the command file by deleting all variables that begin with the command file name preceded and followed with underline characters.

The above example uses the FINFO function. FINFO allows you to find out 44 different things about a file, such as the EOF, if the file exists, the creator, various dates, the size, etc. Appendix B of the Commands Reference Manual gives a complete description of the FINFO function.

FINFO has two arguments. The first one is a character string. In this example the first argument of FINFO is a command file parameter and it must be enclosed in quotes and preceded with the exclamation mark. (We'll look at other kinds of arguments in later examples).

The second argument of FINFO determines the data to be returned. This argument may be either an integer number or a character string describing the data. I much prefer the character string because it documents the command file better. However some of the character strings are obnoxiously long and it's easier to put in a comment than it is to use the designated string.

Prompting for Parameters

Another effective technique is to prompt the user when parameters are not entered. For example:

```
PARM file=' '

IF '!file' = ' ' THEN
  INPUT _find_file;PROMPT='Enter a file name ';WAIT=60
ELSE
  SETVAR _find_file '!file'
ENDIF

SETVAR _find_creator FINFO(_find_file,'CREATOR')
LISTFILE !_find_file, QUALIFY
ECHO Creator is  !_find_creator

DELETEVAR _find_@
```

Prompting may be used either as an alternative to, or in addition to, command file help. If the user enters a file name the command continues. If not, the user is prompted to enter a file name.

Notice the "WAIT" parameter on the INPUT command. This will cause the command to wait for input, for up to sixty seconds in this case, and then continue execution if the input is not entered.

In this example the first argument of FINFO is a variable that has been set to a string and it is not necessary to use either quotation marks or an exclamation point.

INPUT from a Flat File

In the next example the INPUT command is used to read from a flat file by redirecting the command input.

```
PARM file

SETVAR _find_creator FINFO('!file','CREATOR')

INPUT _find_line < !file

LISTFILE !file, QUALIFY
ECHO Creator is  !_find_creator
ECHO First line is
ECHO !_find_line

DELETEVAR _find_@
```

When used this way, the INPUT command reads only the first line of the file. No matter how many INPUT commands you execute, you will always read the first line of the file.

I have found this feature useful for our FORTRAN source files. FORTRAN 77 has a VERSION directive that allows you to include a text string in your code that is carried through to the object code. The text string in the object code may be displayed using the VERSION program (in PUB.SYS). We put a $VERSION directive as the first line of all FORTRAN source files. The text string includes the file name, version number, date, programmers name and destination of the object module (ie. XL, NMRL, or program). We have a single command file for compiling that reads the $VERSION statement, compiles the code and then, based on the $VERSION text string, does the LINKEDIT step to add the module to the appropriate XL, RL or link it as a program. Using the same command for all compiling greatly simplifies compiling and the VERSION information documents both the source and object files.

INPUT from a Message File

As each line is read from a message file it is removed and the following line becomes the first line. So, although the INPUT command will only read the first line of a flat file, it can be used in a loop to read all of the lines from a message file.

This example does a LISTFILE to a message file and then reads the message file to get the names of each of the files in the file set. This provides a customized LISTFILE, providing you have sufficient capability to access the file creator information.

```
PARM fileset

FILE FINDTEMP;MSG
LISTFILE !fileset,QUALIFY > *FINDTEMP
RESET FINDTEMP

WHILE (FINFO('FINDTEMP','EOF') > 0) DO
  INPUT  _find_file < FINDTEMP
  SETVAR _find_file RTRIM(_find_file)
  SETVAR _find_creator FINFO(_find_file,'CREATOR')
  ECHO !_find_file  !_find_creator
ENDWHILE

DELETEVAR _find_@
PURGE FINDTEMP,TEMP
```

In the first use of FINFO, to determine the number of records in the LISTFILE file, the first argument of FINFO is the filename and it is enclosed in quotation marks. In the second use of FINFO, to determine the creator, the argument is a variable and is not enclosed in quotes.

Notice also that it is necessary to use the RTRIM function to trim blanks from the right end of the file names read from the message file. This is because the INPUT from the file is a 128 character string, but the FINFO function can only accept a file name of 1 to 28 characters.

Style

I want to take a minute to point out some more elements of style in the above example. I use parenthesis around the WHILE statement conditions. This is not required by the CI, it's just a preference of mine, probably from my FORTRAN background. I also indent the WHILE statement logic. That's pretty standard in most languages. The "DO" keyword in the WHILE statement is optional. I use it for clarity. Lastly, I always clean up after myself by deleting variables and purging any files that were created.

INPUT from a Flat File

A few paragraphs back I said that the INPUT command would only read the first line of a file. Well, there is a way to use it to read the other lines too. Consider the following two example command files.

The first command file, FIND, does a LISTFILE to a temporary file, FINDTEMP, and then executes the second command file, FINDIT, telling it to read its input from the FINDTEMP file.

Example command file FIND.CMD

```
PARM fileset=@

LISTFILE !fileset,QUALIFY > FINDTEMP
XEQ FINDIT < FINDTEMP
PURGE FINDTEMP,TEMP
```

The second command file, FINDIT, has no parameters. It is called from the other command file and obtains its input from the FINDTEMP file. It determines how many records are in the FINDTEMP file, and then loops using the INPUT function to read one record at a time from the file.

Example command file FINDIT.CMD

```
SETVAR _find_I 1
SETVAR _find_eof FINFO('FINDTEMP','EOF')

WHILE (_find_i <= _find_eof) DO
  SETVAR _find_file INPUT()
  SETVAR _find_file RTRIM(_find_file)
  SETVAR _find_creator FINFO(_find_file,'CREATOR')
  ECHO !_find_file  !_find_creator
  SETVAR _find_i _find_i + 1
ENDWHILE

DELETEVAR _find_@
```

This example accomplishes exactly the same thing as the previous one except that it executes approximately three times faster. I suspect that the speed is due to the fact that the message file system is still (as of 4.0) in compatibility mode. The main disadvantage of this technique is that it requires two command files.

Entry Points

This next example accomplishes the same function as the last example except that it only requires one command file. It uses a technique commonly called "entry points" to provide two separate sections of code in the same command file. An "entry" parameter is tested to determine which section of the code is to be executed.

```
PARM fileset=@
PARM entry=MAIN

IF ('!entry' = 'MAIN') THEN
  LISTFILE !fileset,QUALIFY > FINDTEMP
  XEQ FIND ENTRY='NOTMAIN' < FINDTEMP
  PURGE FINDTEMP,TEMP
ELSE

  SETVAR _find_i 1
  SETVAR _find_eof FINFO('FINDTEMP','EOF')
  WHILE (_find_i <= _find_eof) DO
    SETVAR _find_file INPUT()
    SETVAR _find_file RTRIM(_find_file)
    SETVAR _find_creator FINFO(_find_file,'CREATOR')
    ECHO !_find_file  !_find_creator
    SETVAR _find_i _find_i + 1
  ENDWHILE
ENDIF

DELETEVAR _FIND_@
```

The first section of code, the "MAIN" entry, does the LISTFILE to the temporary file, FINDTEMP. It then executes the same command file again passing "NOTMAIN" as the entry argument. This time the second section of code executes to find the number of records in the FINDTEMP file. It then uses the INPUT function to read the records of FINDTEMP. When this section of code is complete it returns and the first execution of the command file resumes execution at the statement following the XEQ, purges FINDTEMP, and deletes the variables.

One disadvantage of the entry point technique is that it requires the name of the file to be embedded in the file. This makes it inconvenient to rename the command file if you don't happen to like the name that it was originally given. We'll look at a solution to this later.

SETVAR Function in WHILE

The next example is the same as the last one except that it shows a slightly different technique for incrementing the WHILE loop counter. It uses the SETVAR function within the WHILE statement to increment the loop counter instead of the SETVAR command at the end of the loop.

```
PARM fileset=@
PARM entry=MAIN

IF ('!entry' = 'MAIN') THEN
  LISTFILE !fileset,QUALIFY > FINDTEMP
  XEQ FIND ENTRY='NOTMAIN' < FINDTEMP
  PURGE FINDTEMP,TEMP
ELSE
  SETVAR _find_i 0
  SETVAR _find_eof FINFO('FINDTEMP','EOF')
  WHILE (SETVAR(_find_i,_find_i+1) <= _find_eof) DO
    SETVAR _find_file INPUT()
    SETVAR _find_file RTRIM(_find_file)
    SETVAR _find_creator FINFO(_find_file,'CREATOR')
    ECHO !_find_file  !_find_creator
  ENDWHILE
ENDIF

DELETEVAR _FIND_@
```

FINFO Function in WHILE

Again this next example accomplishes the same function as the last few. It combines the SETVAR and FINFO functions into the WHILE statement. I have included this example because I have seen this technique used by others, not because I condone it.

```
PARM fileset=@
PARM entry=MAIN

IF ('!entry' = 'MAIN') THEN
  LISTFILE !fileset,QUALIFY > FINDTEMP
  XEQ FIND ENTRY='NOTMAIN' < FINDTEMP
  PURGE FINDTEMP,TEMP
ELSE
  SETVAR _find_i 0
  WHILE SETVAR(_find_i,_find_i+1) <= FINFO('FINDTEMP','EOF')
    SETVAR _find_file INPUT()
    SETVAR _find_file RTRIM(_find_file)
    SETVAR _find_creator FINFO(_find_file,'CREATOR')
    ECHO !_find_file  !_find_creator
  ENDWHILE
ENDIF

DELETEVAR _FIND_@
```

It is possible to combine multiple functions together in single statements. This technique will shorten command files, however, the statements begin to become difficult to understand. Also, in this particular example, the statement becomes less efficient because the FINFO function is re-executed each time the loop is executed. This is less efficient than determining the number of records before starting the loop.

Handling Error Conditions

Up to this point, for simplicity, I have ignored testing for error conditions in the example command files. What if files do not exist or there are no files in the file set, etc. Good command files test for all possible error conditions, and good coders test command files for all possible errors.

For example, in one of our previous examples we could use the FINFO function to determine if a file exists with something like:

```
PARM fileset

LISTFILE !fileset,QUALIFY > FINDTEMP

IF (FINFO('FINDTEMP','EXISTS')) THEN
  ...
ELSE
  ECHO
  ECHO No files in the fileset !fileset
  ECHO
ENDIF
```

or we could clear and then test the HPCIERR variable to determine if the file exists. i.e.

```
PARM fileset

ERRCLEAR
LISTFILE !fileset,QUALIFY > FINDTEMP

IF (HPCIERR = 0) THEN
  ...
ELSE
  ECHO
  ECHO No files in the fileset !fileset
  ECHO
ENDIF
```

The ERRCLEAR command clears the variables CIERROR, HPCIERR, HPCIERRCOL, and HPFSERR.

Fancy Error Messages

You can add highlighting to your error messages to make then stand out better with something like:

```
ECHO
ECHO ![CHR(27)+"&dB"] No files !fileset ![CHR(27)+"&d@"]
ECHO
```

![CHR(27)+"&dB"]   turns on highlighting and
![CHR(27)+"&d@"]   turns it off.

If you have several different error messages that require highlighting you can set variables for the enhancement strings. This makes the files easier to read, not to mention, easier to type.

```
SETVAR ON  CHR(27)+"&dB"
SETVAR OFF CHR(27)+"&d@"

ECHO
ECHO !on No files in ![UPS("!fileset")] !off
ECHO
```

In this example the UPS function is used to upshift the input argument, a technique that helps to emphasize the error output. An error message from this command might look like:

```
No files in X@.CMD
```

Other Error Conditions

Another type of error condition that needs to be accounted for is the deletion of variables when no variables have been set. For instance, the following command:

```
DELETEVAR _find_@
```

would result in the message:

```
"NO MATCH FOUND FOR THIS VARIABLE SET"
```

if there were no variables in the variable set. This message would be meaningless and annoying to the command file user. To suppress this type of message, since you really don't care anyway, you can redirect the output of the command to $NULL. i.e.

```
DELETEVAR _find_@  > $NULL
```

Choose Your Own Command Name

If you execute the command "!-1" at a colon prompt it is equivalent to the "DO" command. The command:

```
SETVAR _my_name "!-1"
```

results in a variable string containing the last command executed. If you put this command as the first command in a command file, the variable is set to the command that was entered to execute the command file.

The following commands can be used to obtain the name of the command file.

```
SETVAR _my_name "!-1"

SETVAR _my_name  UPS(_my_name)

IF POS("XEQ ",_my_name) > 0 THEN
  SETVAR _my_name  _my_name - "XEQ "
ENDIF

SETVAR _my_name LTRIM(_my_name)

IF POS(" ",_my_name) > 1 THEN
  SETVAR _my_name LFT(_my_name,POS(" ",_my_name)-1)
ENDIF
```

These commands upshift the input, strip off "XEQ" (if it was used), strip off leading blanks, and strip off any arguments that were entered, leaving a variable with the name of the command file.

Now, looking back at the multi-entry point FIND command file (page 14), we can replace the line:

```
XEQ FIND ENTRY='NOTMAIN' < FINDTEMP
```

with:

```
XEQ !_my_name, ENTRY="NOTMAIN" < FINDTEMP
```

resulting in the following code:

```
...
IF ('!entry' = 'MAIN') THEN
  LISTFILE !fileset,QUALIFY > FINDTEMP
  XEQ !_my_name, ENTRY="NOTMAIN" < FINDTEMP
  PURGE FINDTEMP,TEMP
ELSE
...
```

Now the command file is independent of its own name. Anyone can rename the file and it will still work. I'll call it "FIND", and you can call it "WHEREIS".

We can even include the following in the help section of the command file to give the proper command name with the help

```
ECHO !_my_name - LISTF a fileset with Creators name
```

Continuation Lines

Just as you can use an ampersand to continue a command beyond one line in a session, you can also use ampersands to continue commands in a command file. Sometimes, when commands get too long or have multiple parallel phrases, it is more readable to continue the commands on multiple lines. For example:

```
PARM what=" "

SETVAR _S_WHAT UPS("!what")

IF ("!_s_what" = " " &
 OR "!_s_what" = "S" &
 OR "!_s_what" = "J") THEN
  SHOWJOB JOB=@!_s_what
  RETURN

ELSEIF ("!_s_what" = "M") THEN
  SHOWME
  RETURN

ELSEIF ("!_s_what" = "O") THEN
  SPOOLF @;SHOW
  RETURN

ENDIF
```

This example also shows the use of the ELSEIF command and multiple commands in a single command file. This is a convenient way to consolidate multiple SHOW commands in one file. I actually include SHOWME, SHOWTIME, SHOWJOB, SHOWDEV, SHOWCATALOG, SHOWOUT, SHOWQUEUE, SHOWVAR, and a few other custom SHOWs in my command file.

When you combine multiple commands like this in a file, it's a good idea to put a RETURN statement at the end of each section. This interrupts the command at the point it is logically finished and spares the CI from interpreting the remainder of the file (and spares the user from waiting around for it to finish).

System Message Spacing

The following is excerpted from CHKDAT.COMMAND.CSLXL, a contributed library command file to check the status of a DAT tape drive.

```
SHOWDEV !ldev > CKTEMP
...
SETVAR _chkdat_avail STR(_chkdat_ckin,11,1)
SETVAR _chkdat_vol   STR(_chkdat_ckin,43,1)
SETVAR _chkdat_den   STR(_chkdat_ckin,57,1)

IF _chkdat_avail = "A" AND _chkdat_vol ="(" AND &
( _chkdat_den = "1" OR _chkdat_den = "6") THEN
```

It checks specific columns (11, 43, 57) of output from the SHOWDEV command. Between MPE/XL 3.1 and 4.0 the format of the SHOWDEV command was changed so the CHKDAT command file no longer worked. You need to be aware of this eventuality anytime you rely on the spacing of MPE output. From what I have read, I expect that there will be changes in the output of numerous commands in release 4.5.

Purging Files

If you issue the command:

```
PURGE !file
```

and the file does not exist, you receive an error message and the command file aborts. However, if you precede the PURGE with a CONTINUE, (or set HPAUTOCONTINUE TRUE) you still get the error message but the command file does not abort.

As an alternative, you could use something like this:

```
SETVAR _find_savemsg HPMSGFENCE
SETVAR HPMSGFENCE 2

CONTINUE
PURGE !file

SETVAR HPMSGFENCE _find_savmsg
```

By setting the HPMSGFENCE variable to 2, the error message is suppressed and the file will continue. However, if the command file is aborted by the user while the message fence is suppressed, it will stay suppressed and the user will not receive any error messages until after he calls you to find out what happened and you have him reset it.

By using FINFO to determine if the file exists, and only executing the PURGE command if it does, you avoid these pitfalls. i.e.

```
IF (FINFO("!file","EXISTS")) THEN
  PURGE !file
ENDIF
```

An even simpler method is to redirect the PURGE command to $NULL.

```
PURGE !file > $NULL
```

This technique works just as well and only requires one line. For some reason, you don't even need the CONTINUE statement.

Debugging

An invaluable aid in debugging command files is the HPCMDTRACE variable. You can set or unset HPCMDTRACE interactively. While it is TRUE it lists every command executed. You need to remember to turn it off when you are finished debugging, although it's pretty obvious when it's on and you execute a command file.

Another debugging technique is to use OPTION LIST and OPTION NOLIST commands. In MPE V, these statements could only be used in the header of a UDC. In MPE/iX they may be used at any point in a command file. Whenever LIST is on, all commands executed are listed. When it is off they are not. By using these commands you can list commands in selected parts of your command file.

For more sophisticated debugging, particularly when you may want to leave the debugging tool in the code but have it transparent to the normal user, you can add a debug parameter to the command file. i.e.

```
...
PARM debug=0

IF (!debug > 0) THEN
  SETVAR HPCMDTRACE TRUE
  SETVAR HPMSGFENCE 0
  SHOWVAR
ELSE
...

IF !debug = 0 THEN
  PURGE FINDTEMP,TEMP > $NULL
ENDIF
```

To execute the command in debug mode, specify the debug parameter by keyword. i.e.

```
:FIND  C@,  DEBUG=1
```

The command file can display selected variables and can even skip file purging when in debug mode.

Documentation

If you expect your users to know and use your command files, you need to document them. In addition to the ECHO help in command files, we have a documentation file for each command. These are kept with the same name as the command in the DOCCMD group.

In addition, our editor, FSEDIT, allows us to put a 64 character description with a file. The description is kept in the first user label of the file. I also added this capability to the CSL program GSCAN. All of our command files have these one line descriptions. We then use a command file, DESCR, that is similar to the FIND examples I have been discussing to display the descriptions.

The help for DESCR is:

```
DESCR - List files in a fileset with Descriptions

Syntax

  :DESCR [fileset] [,find-string]

Parameters

  fileset     - The HP fileset; default @.CMD
  find-string - A string to search for
```

Example output from the DESCR command

```
Files in the fileset A@.CMD with FSEDIT Descriptions

A        - ABORT command file
ACCTINFO - Formatted output of System Accounting Structure
ADDR     - Print address labels
AJ       - ABORTJOB
AS       - Alter spool file PRI to 12 so it will print
```

DESCR.CMD

```
PARM fileset="@.CMD"
PARM find =" "
PARM entry=MAIN

IF "!fileset" = "?" then
  SETVAR S,CHR(14)
  SETVAR B CHR(14)+':'+CHR(15)
  ECHO ![S+'R,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,T'],
  ECHO ![B+'   DESCR - List files in a fileset with Descriptions       '+B]
  ECHO ![S+'@99999999999999999999999999999999999999999999999999999999999[']
  ECHO ![B+'                                                           '+B]
  ECHO ![B+'   Syntax                                                  '+B]
  ECHO ![B+'     :DESCR fileset [,find_string]                         '+B]
  ECHO ![B+'                                                           '+B]
  ECHO ![B+'   Parameters                                              '+B]
  ECHO ![B+'      fileset     - The HP fileset; default @.CMD          '+B]
  ECHO ![B+'      find_string - a string to search for                 '+B]
  ECHO ![B+'                                                           '+B]
  ECHO ![S+'F;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;G']
  DELETEVAR S,B
  RETURN
ENDIF

IF '!entry' = 'MAIN' THEN
  ERRCLEAR
  SETVAR _descr_find '!find'
  LISTFILE !fileset,QUALIFY > DESCRTMP
  SETVAR _descr_i 0
  IF HPCIERR = 0 THEN
    SETVAR _descr_eof FINFO('DESCRTMP','EOF')
    ECHO Files in the fileset !fileset with FSEDIT Descriptions
    ECHO
    XEQ DESCR ENTRY='NOTMAIN' < DESCRTMP
  ELSE
    ECHO File !fileset NOT found
  ENDIF
  PURGE DESCRTMP,TEMP
  DELETEVAR _descr_@

ELSE
  WHILE SETVAR(_descr_i,_descr_i+1) <=_descr_eof DO
    SETVAR _descr_file INPUT()
    COMMENT Does it have User Labels Written?
    IF FINFO(_descr_file,10) > 0 THEN
      COMMENT Yes, read the fisrt user label
      SETVAR _descr_file_descr RHT(FINFO(_descr_file,25),64)
      IF LFT(_descr_file_descr,1) < " " OR  &
         LFT(_descr_file_descr,1) >= "}" THEN
        SETVAR _descr_file_descr RPT(".",64)
      ENDIF
      SETVAR _descr_len 9-POS(".",_descr_file)
      SETVAR _descr_file_descr RTRIM(_descr_file)+ &
      RPT(" ",_descr_len)+" - "+_descr_file_descr
      IF _descr_find <> " " THEN
        IF POS(UPS(_descr_find),UPS(_descr_file_descr)) <> 0 THEN
          ECHO !_descr_file_descr
        ENDIF
      ELSE
        ECHO !_descr_file_descr
      ENDIF
    ENDIF
  ENDWHILE
ENDIF
```

References

MPE XL Commands Reference Manual 32650-90003

CI Access and Variables Programmers Guide 32650-90011

INTERACT	**Page**	**Title / Author**
Nov 88	44	New Features of MPE XL User Interface Thomas Shem & Jeff Vance
Sept 89	26	Life of an MPE XL Command Scott Cressler & Jeff Vance
Mar 90	96	Programming with MPE XL Heikki Tsakinen
Feb 91	83	Advanced CI Programming, The 2.1 Story Scott Cressler, Jeff Vance, Steve Elmer
Sept 91	86	MPE XL Command Files Fred Ochs, Jr.
Nov 92	26	Creating Command Files Rene Martinez
Nov 92	78	What the Manuals Don't Say John Dunlop

Paper No. 5004
RF/DC and the HP 3000

Joe Howell
Professional Products, Inc.
PO Box 589
Defuniak Springs, FL 32433

What Is RF/DC and How Does It Work?

The acronym RF/DC stands for Radio Frequency Data Communications. RF/DC encompasses the hardware and software technologies used to transmit data between a hand held computer with an integrated VHF radio and a base station connected to some kind of host computer system. Communications software in the hand held unit passes the data to the radio for transmission. The radio in the base station receives this data and passes it via a serial or LAN connection to the host computer.

What are the Hardware and Software Components?

Each vendor in the RF/DC market has a different implementation of this technology. This paper will present a general overview of the hardware and software components of an integrated RF/DC system and their desirable (and some undesirable) features.

Hardware:

Let us start with the **hand held unit.** This device is a portable, programmable stand alone computer. Ideal attributes include :

LCD display with adjustable contrast, back lighting, and a minimum of 16 lines by 21 characters width

Integrated laser scanner - Many vendors do not integrate the scanner, but let you attach the device of your choice via a serial port. We feel that the integrated scanner is a worthwhile feature because this gives you a single hand operation.

Pistol Grip - this usually houses the laser trigger mechanism, battery packs, serial ports, and recharger circuitry. External high speed recharges should be available for high usage applications.

Programmable Keypad. The keypad should be full alpha, numeric, and function key keypad. This should be configurable by the application software. Cursor positioning keys are also an advantage.

VHF Radio - The 2 types of VHF radios used in hand held units are known as Narrow Band Radio and Spread Spectrum Radio. The relative merits of each of these will be discussed later.

Flash EPROM Memory. This is memory that is used to hold the hand helds' operating system, radio and serial device drivers, terminal emulators, and application programs if this is a user programmable device.. Flash EPROM is a real improvement over earlier ROM's where you had to remove the chip and send it to the vendor for new operating systems, drivers, or applications to be loaded. Most flash EPROM's can be loaded by any PC using a special PC program and a direct connect download cable. We found that 128K was plenty of flash memory to hold this system software.

RAM. How much can you afford? Most hand helds come with 128K. We have several working with 256K. Most vendors offer up to 1MB of regular memory. The amount required depends on the amount of data to be collected and stored before being forwarded to the host.

Next lets consider the **Base Station.** This device is used to provide the VHF communications between the hand held and the host. Some base stations are presented as "Black Boxes" with a serial or LAN connection to the host. Many vendors call the base station their **Communications Gateway.** When you look under the hood, you find that most of them are simple PC's running DOS or, in some cases, a real-time operating system. These DOS machines are running a communication program which talks to the internal VHF radio board for the RF traffic and to a serial port or LAN board for the host connection. More will be said about these communication programs later. The main consideration about the hardware is to insure that the vendor is using a FAST CPU and plenty of memory. Frequently, poor RF performance is due to the vendor low-balling the CPU and memory inside the "black box" base station.

If your facility is large (a subjective term, I know, but lets say over 200,000 square feet) you may not have good RF transmission due to interference from walls, equipment, and stored goods such as paper and fabric. The RF vendors have solved this by offering **cellular repeaters**. These are units which receive an RF transmission and re-transmit it. The re transmission media may be over a hard wired backbone to another repeater or the base station , or back on the RF airways, much like a cellular telephone network.

The final piece of hardware is the host computer. In our search for turn-key RF/DC systems, we found that a majority of the vendors and systems integrators wanted us to use a PC as the host computer. Their base station would establish a serial connection to the PC and upload or download files or records from PC based application. It would be up to you to develop the application to move the data to and from your corporate host computer. We chose to have our host be our production HP 3000, and not to include a PC as a middle store-and-forward device. The primary reason for this was our need to move up-to-the-minute information from the HP host to the hand held computers and to update our host based production order and manufacturing systems with data collected from the hand helds as events happened around the plant. The PC-in-the-middle scenario add a level of complexity and communication delay which did not fit in to our goals of distributing data to the users of the hand held computers as rapidly as possible.

Software

The software components required to make RF/DC work are as important as the hardware. They determine what options are available to you to solve your specific RF/DC application needs. Fortunately, there are several options available. Starting in the hand held computers, lets consider these options.

There are three ways the hand held can function in the RF/DC world. The first is that a hand held can run a terminal emulation program. The obvious benefit of this is that the host application programs would think they are working with an existing terminal and no reprogramming would be needed. This also means that your application programmers don't need to learn another language. Several vendors offer this option for IBM host systems. Using 3270 and 5251 emulation, they have a true plug and play solution. In the HP 3000 world, I am only aware of one vendor that currently emulates 700/92 block mode terminals, and this implementation is via an X.25 PAD, not an async or LAN connection. The mapping of the block mode screen to the hand held screen takes place in the vendors' base station/gateway via a cut and paste type of interface.

The second way a hand held can function is as a stand alone terminal responding to commands and prompts from a host program. These prompts would control the screen, energize and configure the laser scanner, serial ports, define local edits, and format the data to be returned to the host program via the RF link. Some sites perceive this as an advantage because all the application code resides on the HP host. The learning curve on these command and control languages is usually short.

The third way for a hand held to function is as a stand alone programmable, portable computer running a series of application programs which have been developed on a PC and loaded in to the flash EPROM. When the hand held application requires information from the host, or needs to pass information to the host to update the database, it transmits a request to a host based communications program supplied by the user, along with the information request or information message. This host based communications program would then run the requested application program passing the information request or message along. When the host application program completes its task, a status or answer message is returned via the host communication program and the RF link to the hand held. While this is by far the most complex of the three environments the hand held can function under, it offers the application designer the most flexible use of the tools available. The hand held is truly functioning in a wireless client server environment. This approach would allow the hand held to request a unit of work from the host, go off-line from the host and process this unit of work, and return the updated work to the host. The actual transmission of data would be as a file of records, or as one long buffer. In either case, the traffic on the RF network is significantly less than either one of the conversational modes above. This means that more hand held units could run concurrently without saturating the capacity of the RF network.

Another software component to consider is the programming language for the hand held, if you choose to use one as a standalone computer. Some vendors with programmable hand held computers allow you to develop your hand held applications on PC's with the language of your choice. Others offer their own languages which have been enhanced to take advantage of the features of their hand helds' operating system. We found that this was a successful approach. We chose a vendor with a COBOL compiler on the PC that also handled all of the device driver calls for the hand held radio, screen, and serial devices. We were able to become productive in this somewhat standard language in a few short weeks.

One area you have very little control over is the communication software in the vendor's base station which communicates with the radio in each hand held. This is an area of major importance. If the vendor has a poor communications package, the throughput of the network will suffer. The only way to get a feel for the true performance level is to talk to the installed base. The customers are usually quick to tell you if they have a performance problem.

VHF Radios - Narrow Band VS Spread Spectrum

The two VHF radio technologies being used in the RF/DC environment are Narrow Band and Spread Spectrum transmission. Narrow Band radios transmit and receive on a preset frequency . These hand held units and the base station require a site specific FCC license, and transmit at a maximum rate of 9600 BPS. Spread Spectrum involves wide band modulation over a frequency range of 26000 Hz at a data transmission rate of up to 300,000 BPS. These radios do not require FCC license. Both radio types are subject to all sorts of RF interference, however, Spread Spectrum throughput is not really affected due to the high data transmission rate. It should be noted that not all Spread Spectrum radios and communication programs in the base stations are equal. If you survey the vendors offering spread spectrum and their customers, you will find a wide variety of performance and throughput rates being claimed. Again, talk to the installed base.

Connectivity & Communication Issues

Connecting the base station you have selected to your HP 3000 is not always a simple event. If you are using an async interface, one would think that a simple DB25 or DB9 to DTC 3 Pin or 25 Pin connection would be all that was required. There are several "GOTCHA'S" here. First, most of the vendors base stations use hardware handshaking for flow control. The serial ports on a DTC or an ATP use software handshaking. They are also extremely dumb. They were designed years ago to handle character and block mode data transmission **primarily to terminals**, not to talk to a very fast PC transmitting large blocks of data. For this reason, you frequently get errors when you do a read-after-write to a serial port from an application program. The solution we have found is to use a custom made "Black Box" from Telamon, Inc. This box talks hardware handshaking to the base station and software handshaking through a separate read and separate write port to the HP host. A lot of the polling overhead is also handled by the Telamon engine.

If you use a LAN connection, other complications arise. Lets say you use TCP/IP as the transport software. This would require a TELENET card in the DTC and the corresponding system software to drive it. One assumes that the vendors communication package and LAN hardware would provide the matching TCP/IP software. Since TELENET is a character mode protocol, you would rule out any block mode terminal emulation.

A third connection option is still through a LAN, but to have the base station support some type of Virtual Terminal (VT) connection. This would require the VT services of the LAN Link or NS3000 product on the host. The base station would need a product along the lines of WRQ Connect 3000 product to handle the virtual terminal communication over the LAN. The hand held would then need a terminal emulation program to complete the package. The obvious advantage of this connection is that no reprogramming would be needed for existing applications. From what I have been able to find out, no vendor has all of these pieces actually in production yet. Techlogix is the closest, but their product uses a X.25 connection.

As you can see from the software and hardware considerations, putting up the infrastructure to run RF/DC is not as simple as the marketing types would have you believe.

Applications Specifics

Lets assume you have selected an RF/DC vendor, have the hardware installed, and have learned the programming language to get data to and from your HP host. Now lets look at what applications are feasible under RF/DC.

The smartest thing we did in our whole RF/DC effort was to pick a very small pilot project to implement first. This had to be a project that would interface with our existing manufacturing and corporate information systems, but one which would not affect customers or production as we tuned it. We chose a finished goods inventory put-away application as this pilot. This application moves goods from the final inspection point to the finished goods shelf, or in the case of a customer order, moves the goods directly to the shipping point. In order to meet the requirements of this simple project, we had to develop the following modules:

1. Host application program to find the production order transmitted, and update inventory quantity on hand, notifying the hand held of any errors.

2. Master Communication program to perform process handling to start Host Application Programs, pass them data, receive returned data, and pass this returned data to the base station.

3. Hand Held application program to solicit bar coded id of products to be put up, send these id's to the base station, send the put away location to the base station, and receive

confirmation that the Host Application Program successfully updated the database.

RF communication and logical application error handling had to be created for each of these programs. The nice thing about our simple pilot project was that, once we had this up and running, all new functions added to the hand helds use outlines of the same programs created for the pilot. This means that each new function we bring on-line has a shorter development and test cycle.

The following application areas are under development in our shop:

Raw Materials Inventory Control: This application includes the receipt, labeling, put away, and picking of all of our raw materials.

Work In Process Inventory Control: One critical area in our manufacturing company is the control of work in process. This project includes not just how much and where our inventory is, but what are we working on . The ultimate idea is for the system to pass the operator of the hand held information to help them to know what is the highest priority work to be done next, and what and where are the in-process pieces needed to do this work.

Paperless Order Picking and Shipping. We call this system **P**ick-it, **M**ove-it, and **S**hip-it, or PMS for short. Over the years we have found that we handle an order several times in getting it out the door. This is primarily due to the nature of our average order, in that it contains some items which are pulled from stock and some items that are made to order. We try to ship all orders complete, so we must manufacture the custom items and match them up with the rest of the order which may have already been picked and is in a "Box and Wait" area. This matching is very error prone. Under the RF/DC system, the order will not be picked until the custom items have already been made and are ready for shipment after inspection. The matching of bar codes on the shipping containers and those on the box end labels by the program in the hand held alleviate the problem of the custom item getting put in the shipping carton of the wrong order. Another big advantage is that the boxing and checking of the order can be done anywhere in the shipping area, not just where there is an HP terminal. This allows us to add extra shipper/boxer/checkers to handle busy times without having to allow extra hard wired terminals.

The payback in each of these application areas comes from several sources. The most noticeable is accuracy. If we are updating inventory by bar code at the time it actually goes in the shelf, the errors from data entry are reduced. If we ship the right product to the right customer, customer satisfaction increases and mis-shipments and their associated costs go down. If we have an accurate inventory, we can schedule production more accurately and in a timely fashion. This means we will spend our time making the items we need when we need them, where we need them, for the customers who need them. Our estimate is that we can improve our delivery and service levels to our customers and reduce total manufacturing cost by 15%. This is a significant payback for any expenditure in RF/DC technology.

What's Next?

RF/DC has changed considerably since we begin our investigations three years ago. LAN connectivity is being developed by most of the vendors. Terminal emulation on the hand held computers is perceived as the fastest way to get an RF/DC application up and running. Integrated laser scanners and attached printers are evolving on the hand held units. Spread Spectrum communication technology is being improved, and wireless cellular repeaters are reaching an affordable price range. On the leading edge, some vendors are offering wrist mounted RF/DC units that look like something The Terminator would deploy. At Disney world, you can order a drink from your golf cart and charge it to your room. In retail outlets around the country, portable point of sale systems with pen interfaces are emerging. The only limit we have seen to where RF/DC may be applied is the limit established in our minds.

RF/DC Vendors

Hand Held Products
8008 Corporate Center Dr
Charlotte, NC 28226
704-541-1380

Intermec
PO Box 360602
Lynnwood, WA 98046-9702
206-355-9700

LXE Corporation
PO Box 92600
Norcross, GA 30092-9200
404-447-4224

Norand Corporation
550 Second St SE
Cedar Rapids, Iowa 52401
319-369-3100

Symbol Technology
116 Wilbur Pl
Bohemia, NY 11716
516-563-2400

Teklogix
7914 Tanners Gate
Florence, KY 41042
800-633-3040

Telxon Corporation
PO Box 5582
Akron, OH 44334-0582
800-800-8001

Publications & Meetings

ID Systems
PO Box 5245
Pittsfield, MA 01203-5245
Helms Publishing Co
603-924-9631

Scan Tech 93
Oct 18-21
Philadelphia, PA
800-338-0206

ID Expo
May of each year
Chicago, IL
Contact ID Systems

HP 3000 Communications

Ross Scroggs
Telamon Corporation
510-210-6864

Doug Walker
WRQ, Inc
Seattle, WA 98102
206-324-0407

Paper # 5005

SOFTWARE QUALITY AND THE HP 3000

Bob Mead

Hewlett-Packard Company
19111 Pruneridge Avenue
Cupertino, CA 95014

(408) 447-6059

INTRODUCTION

Software quality has always been important to customers -- after all, who isn't in favor of quality. In the past several years however, the consistent delivery of high quality and extremely reliable software has become of paramount importance to many HP 3000 customers.

Several factors have influenced this dramatic increase in the overall importance of software quality. Many HP 3000 systems are now used in "mission critical" business applications, where the company's business literally comes to a standstill if the system is unavailable. The processing power of an HP 3000 has also increased substantially, so that having hundreds of users connected to a single HP 3000 is now common. Finally, recovery time after a system abort for these large configurations has unfortunately increased, further magnifying the impact of a system abort. The net effect of these factors is that the cost of a failure to the customer has grown significantly, in terms of lost productivity, lost sales, customer satisfaction, and/or manufacturing delays. In some cases, the cost to a company for a single event can be in excess of one million dollars.

Poor software quality also has a negative impact on the vendor as well, in this case Hewlett-Packard. Software quality problems obviously can (and do) lead to lower customer satisfaction, and in today's extremely competitive computer industry it is extremely important to achieve and maintain very high levels of customer satisfaction. The cost of correcting software problems once the product has been delivered to customers is also quite high, making it more difficult to invest in activities that further enhance the overall product set.

Simply detecting defects before the product is shipped to customers is not sufficient in today's environment either (assuming that it ever was). Finding and fixing defects in complex environments requires considerable effort, and also substantially increases time-to-market for a product. It has become much more important to "do it right the first time", since the productivity improvements and cost savings of preventing defects during the design and implementation phases can benefit an organization in many ways.

Hewlett-Packard has undertaken a number of efforts over the past several years to improve the quality of software delivered for the HP 3000. These include practices that reduce the number of defects introduced during design and implementation, increase the effectiveness of testing done after the implementation phase, and increase the responsiveness to problems encountered by customers.

This paper is primarily focused on efforts to improve the quality of the MPE/iX operating system software, but it is also indicative of efforts throughout Hewlett-Packard to improve the quality of all software products. Before describing the quality improvements however, we will provide an overview of the process used to develop, integrate, and test the roughly 5 million lines of source code which comprise an MPE/iX software release.

MPE/iX SOFTWARE DELIVERY PROCESS OVERVIEW

In an ideal or perfect world, the process of delivering an MPE/iX software release would be quite simple, consisting of the following steps:

1) Developments teams (lab) design, implement, and test products for the release
2) Individual products integrated to create the system release tape
3) Regression/Certification testing of the release as a whole
4) Manufacture and distribute to customers

	Software Dev Orgs	System Integration	System Test	Non-factory HP Orgs	HP Software Manufact'ng	Customers & VABS
Product Development Phase	Source Mgmt Design Implement Unit Test Product Integration Regression Test	Define and distribute integration environment toolset	Plan for Back-end Testing Identify Alpha & Beta Sites			
Start of Backend Process Phase	Submit Object Code to System Integration	Create initial version of S/W Release (System Build)				
Factory Testing Only Phase	Quality Review of System Build Repair of Defects Roll-in of Patches	Distribute builds to partner organization Create new system builds (as required)	Reliability Test (Stress) Power Fail Recovery Configuration Installation & Update			VAB Prep
User Testing Phase	Regression Testing (as needed) Repair of Defects Roll-in of Patches	Distribute builds to partner organization Create new system builds (as required)	Repeat tests as required (based on nature of defects repaired)	Alpha testing - Dev system - Production Provide feedback		Beta testing - Dev system - Production Provide feedback
MR Phase	Final Roll-in of Patches	Create set of Tape Masters (Final build)	Regression Testing		Manufacturing Testing Factory Pre-loading Original S/W shipments S/W Update shipments	Purchase products Request S/W updates Install S/W updates

Activities During an MPE/iX Software Release
Figure 1

In the real world, or at least the one we live in today, Steps 2 and 4 listed above are in fact quite straightforward, and are typically performed with few difficulties. Step 1, as many of you are aware, is far from a simple task, although the description shown does represent the basic steps in the process. In this paper, we will not be addressing the design and implementation phases, other than to describe practices used to minimize the number of defects found in both the product level and system level testing phases.

In order to ensure that Hewlett-Packard delivers high quality MPE/iX software releases, Step 3 is much more complex than the simple statement above. It involves several different testing phases and a number of rework cycles. Figure 1 depicts a simplified representation of the activities performed by various organizations (both internal and external to HP) during the development and delivery of an MPE/iX release. The rows represent the phases that take place over time, with the activities performed by organizations listed in each column. The many activities shown in the rows "Factory Testing Only" and "User Test" comprise what really happens in Step 3.

We will now describe each of the phases in more detail, and then discuss efforts under way in Hewlett-Packard to improve the overall quality of MPE/iX software releases.

DEVELOPMENT PHASE

An integrated MPE/iX software release for the HP 3000 typically contains around 125 products, representing around five million uncommented source code statements (the operating system represents approximately 40% of this total). Application software, such as MM/3000, is not part of the integrated release. Several different organizations within Hewlett-Packard are responsible for these software products, with each organization having ultimate responsibility for the quality of the products it provides.

Each organization is responsible for the following activities during the development phase:

- maintaining its own source code
- generating defect repairs to its code (including patches)
- design, implementation, and testing of the product
- testing new versions of a product with previous versions of other products
- specifying quality criteria for achieving Manufacturing Release (MR)

During the development phase, the code to be included in the next release is designed, implemented, and tested. This code may include new products, enhancements to existing products, and defect repairs. Where there are known interdependencies between products, the new version of each product is tested with the previously released version of other products. For example, during the development phase of Release 4.0, the operating system lab tested the new version of MPE/iX with the Release 3.1 versions of networking products, and the networking labs did just the opposite. This product level testing also includes the execution of a regression test suite to ensure that all functionality continues to work as expected.

In some cases, it is not possible to test the new version of one product with the previous version of another. This typically occurs when a data structure used by both products changes, but can occur for other reasons. In these situations, the involved organizations will jointly develop plans to ensure that the new versions of both products are tested together.

At the completion of the development phase, each product is expected to have met its quality criteria for product MR, and thus should be ready for shipment to customers. The criteria always includes some level of product testing, and may also include beta testing at customer sites. The specific types of tests comprising the product tests depends on the nature of the product. Functionality certification tests are always included, and for products such as the operating systems and networking, stress tests involving execution of the product(s) under heavy load for a specific number of hours (in the range of 72-120 hours) are included.

The length of time in the development phase also varies, with times as short as one month for smaller products, and as long as 12-24 months for large development efforts. In the case of the operating system, a typical release will include several large development efforts, as well as many smaller features and hundreds of defect repairs.

During this phase, planning is done by organizations involved with the release in its latter phases. The system testing organization, for example, will analyze the new content of the release to determine what changes need to be made in the system-level tests. The organization responsible for user level testing begins to recruit test sites for both HP-internal alpha testing and external customer beta testing. A strong attempt is made to identify sites that will test the new functionality being provided in the release.

START OF BACK-END PROCESS

At some point during the development phase, a date is agreed upon as the start of system level activities. This milestone, internally referred to as TC1 (Test Cycle 1), denotes the beginning of the "back-end" of the release schedule (the development phase is referred to as the front-end). Early releases of MPE XL (e.g. Release 2.0) were planned with multiple test cycles at the system level, and additional new functionality was incorporated at most test cycles. Starting with Release 3.1, a single test cycle model was established, but the TC1 designation has remained. At the same time the TC1 milestone date is established, the content of the release is also frozen (except for defect repairs). Subsequent changes to the content must use a formal change control process which assesses the business need and risk associated with the change, and ensures that all organizations involved with the release are notified of the change.

At the TC1 milestone, each organization submits object code for each of its products to the System Integration team. An internal database, called BSTORE, contains the description of how the product is to be integrated in the final release (e.g. group.account for each file comprising the product). Changes to BSTORE (if any) are also submitted at this time.

At this point, the system integration team performs the initial build of the entire release. If no integration problems are encountered, this process takes less than one day. Given the large number of products involved however, it is not uncommon for at least one product to incur some type of problem, requiring multiple integration attempts. A successful integration is usually completed in less than 2-3 days. The cycle of finding problems, fixing problems, and reintegrating the release now begins in earnest.

FACTORY TESTING

As can be seen in Figure 1, there are many activities which occur in this phase, and the number of organizations involved begins to increase. While the chart in Figure 1 would indicate a clear delineation between this phase and the User Testing Phase, in reality the line is somewhat blurred. There are specific criteria established for the starting of user testing, which in some cases are less stringent than the criteria for achieving MR of the software release. This implies that factory testing will continue during the user testing phase.

There are two basic types of testing activities which occur at this point: system level testing and quality review. Quality review is performed by each development organization, and consists of running product specific tests for the products for which the organization is responsible. The purpose of quality review is twofold: first, to ensure that the code submitted by the development organization was correctly integrated into the software release, and second, to ensure that the product correctly functions with the other new or modified products in the release. This testing is performed by the individual development organizations, rather than a centralized testing organization, because we believe that the organization responsible for the product has the most knowledge and experience about how the product is expected to function. It should be noted that most of the development organizations contain dedicated testing teams. An additional benefit of having the development organization conduct the quality review is that equipment can be leveraged for both development phase product testing and system level quality review.

In addition to the quality review testing performed in each development organization, there is system level testing performed by a central system testing group. This testing is focused primarily on the operating system, basic networking, and database software. Four basic types of tests are performed: reliability (stress), power fail recovery, configuration, and installation/update. While not shown in Figure 1, there is also performance testing done by a separate team utilizing some standard benchmarks, with the purpose of ensuring that overall performance does not degrade unexpectedly.

The system reliability tests consist of two HP 3000 systems connected in a network, each being driven by a separate system that is emulating interactive users. The number of emulated interactive users depends upon the power of the processor being tested, with several hundred users being typical for high-end systems. The tests are typically executed on several pairs of systems, including both uni-processor and multi-processor systems. The tests are considered to have passed when each pair of systems runs for 120 hours without a system abort, hang, or critical error. Detected problems that do not result in a system abort or hang are automatically logged by the tests. During the course of the back-end activities for a release, the 120-hour criteria is typically met many times. The organization responsible for networking software also runs its own reliability stress tests encompassing a much wider range of networking products.

Power fail recovery tests are used to accomplish two objectives: first, to ensure that the power fail recovery feature of the HP 3000 works correctly, and second, to thoroughly exercise the interrupt code in the operating system. The tests involve placing a heavy load on the system, and then automatically disrupting power at varying intervals and for varying lengths of time. Some tests interrupt power for the entire system, and others only for either the SPU or the disks. There is a specific set of tests which must be completed, and executing the entire set takes 1-2 weeks. As noted above, these tests frequently find timing problems not specifically related to power fail recovery.

Configuration tests are used to verify that the system functions correctly at both minimum and maximum configuration limits, including both table and device limits. These tests are also automated, and involve placing a heavy load on the system for each of the configurations being tested.

Install and update tests are used to ensure that the new software release can successfully be installed on new systems, as well as installed as an update for releases that may be installed on existing systems. These tests are also used to verify the accuracy of the installation/update documentation instructions.

Once a predetermined subset of the above tests have been passed, the software is made available to any of our VAB (Value Added Business) partners who desire to ensure that their product will function correctly on the new release. For Release 4.0, almost 40 third party companies chose to participate in this program. Special emphasis is placed on third party software to be used by the alpha and beta sites that will be testing the release during the user testing phase.

Throughout this phase, all defects discovered are tracked and reviewed on a daily basis. Those defects which must be fixed are designated as showstoppers, and these must be corrected prior to MR (and in many cases, prior to the start of user testing). The decision on whether or not to repair non-showstoppers defects is made by each development organization, dependent upon the nature of the defect and its impact to customers, the engineering effort required to develop and test the fix, and the risk associated with the fix.

The release is rebuilt on an as needed basis to incorporate defect repairs for the showstopper problems, as well as other defect repairs each organization chooses to include. In addition, patches which have been recently created for releases already installed at customer sites will be incorporated during this phase. As each new build is created, tests which have previously passed may be executed again, depending upon the nature of the changes being made. Some level of reliability testing is always done, although for minor changes only 24-hours of testing may be required (assuming the 120-hour criteria was met on an earlier version of the release).

USER TESTING PHASE

At this point, the release gets it first exposure in environments where the users are trying to do real work. The first system on which the software is installed is frequently a large system which provides HPDeskManager support for most of the development engineers responsible for the MPE/iX operating system, as well the entire management team responsible for the HP 3000. As you might imagine, significant problems will be quickly addressed.

Each release is tested on a number of HP internal alpha sites, primarily in production environments. Sites are selected such that the combined set of systems will use as much of the new functionality in the release as possible. Systems supporting Hewlett-Packard Corporate headquarters are included, as well as systems used by the Response Centers. Each site is expected to provide feedback at least once per week, and more often if problems are being encountered. Sites are requested to install patches and/or new versions of the release as defects are corrected. For Release 4.0, over 15 internal production HP 3000 systems at ten separate sites were used as Alpha sites, with an additional 13 support machines also testing the release.

After the release has been running successfully at the alpha sites for several weeks, the release is shipped to external customer beta sites. Many beta sites will initially install the software on development systems prior to using it on production systems, but some do start with production systems immediately. The number of sites involved is a factor of the amount of new functionality in the release and the number of customers interested in participating as a beta site. As is the case with alpha sites, the selection of sites is done in such a way as to maximize exposure of the new functionality in the release.

Status at beta sites is monitored on a regular basis, and sites are requested to supply a written report at the end of the beta test period. Sites are provided with patches for problems impacting them, and are requested to install the "final" version of the release prior to the MR of the release by the factory. In addition to verifying the reliability and quality of the release, beta sites provide useful feedback on the documentation associated with the release, and exercise the normal post-release support processes within Hewlett-Packard.

The number of beta sites varies from release to release, with ten sites participating in the Release 4.0 beta program, representing a total of 16 systems. The period of time between the start of beta testing and the MR of the release is typically in the range of 6-12 weeks, dependent upon the number of problems found which must be fixed.

Factory testing does continue in parallel with exposure of the software at alpha and beta sites. Each time it is necessary to rebuild the release, a subset of the tests are run as a regression test, with the specific tests being determined by the nature of the changes. As is the case during the factory testing phase, patches generated for previous releases are incorporated into each build, in addition to the defect repairs for problems encountered during release testing. Defects are tracked during this phase in the same fashion as the earlier phase.

MR PHASE

Once it has been determined there are no outstanding showstopper problems and the feedback from beta sites indicates the release is ready for general distribution to customers, intense preparations begin for the MR of the release. The final version of the release is built, using the official VUF designated for the release. The final roll-in of patches is included in the build, with the goal of minimizing the number of patches available for prior releases that are not included in the new release. Note that this number will seldom be zero from the customer perspective, since the elapsed time between this final build and the start of shipments to customers is typically 4-5 weeks. At the time Release 4.0 began shipping to customers, the number of available patches not integrated into Release 4.0 was less than ten.

Once the final build is complete, a final regression test is executed and the master set of tapes is delivered to the Software Manufacturing organization within Hewlett-Packard. A series of manufacturing tests are run to ensure that customized subset tapes can be correctly produced, and then copies of the release are shipped to the response centers and all other organizations within Hewlett-Packard which provide customer support for the HP 3000. Once this distribution is complete, shipments to customers will begin for new systems and software updates.

The manufacturing verification and distribution of the release to support organizations typically takes two to three weeks. During this period the release is still being used by both alpha and beta sites, so should a new showstopper problem be encountered, it will be necessary to rebuild the release and repeat the above steps.

QUALITY IMPROVEMENT EFFORTS

In the past, the majority of the efforts to improve the quality of software on the HP 3000 were centered around increasing the scope and effectiveness of the tests used to certify the software releases. Consistent with this approach, thousands of individual functional validation tests were developed for the early releases of MPE XL, as well as many of the other test suites still in use today. While these tests were instrumental in identifying many defects in the software which were corrected prior to shipment to customers, they were not sufficient to guarantee the delivery of a high quality software release (as anyone who used the very early versions of Release 1.0 would attest).

There are two fundamental problems with placing the major emphasis on defect detection after the implementation is complete:

1) Many defects are not detected, largely due to the fact that functional tests tend to be run in a standalone environment, leaving many of those defects that are dependent upon the configuration, timing, the phase of the moon, etc. to be discovered by customers. It is also not practical to run the very large number of functional tests required in all potential environments, since the number of permutations and combinations is astronomical.

2) Testing after implementation is complete is a very inefficient way to detect and debug problems. Many problems found this way are difficult to consistently reproduce (and hence difficult to diagnose), and a large number of people are required to undertake a massive testing effort, especially when

the tests must be executed many times. Based on data collected during Release 4.0, we estimate that problems found during product level testing of the operating system and system level testing required an average of about 180 hours of engineer effort per defect, simply to detect the existence of a problem.

During the past several years, we have invested considerable resources in detecting defects much earlier in the development cycle. Those efforts are described below, along with other improvements we have made to reduce the number of defects that are present in an MPE/iX software release when it begins to ship to customers.

NO REWORK OBJECTIVE

Many of the efforts were the result of an overall objective that was established several years ago in the operating system development organization. As was noted earlier in this paper, in an ideal world the software release process backend would consist of a single certification test cycle, lasting less than a month, followed by MR and shipment to customers (user testing is not required in this ideal world since the software contains no defects at the point when user testing would normally begin). The fact that the backend takes much longer is the direct result of many rework cycles being required to achieve the desired level of quality.

To address this situation, we adopted a goal of reducing rework. Furthermore, to make the goal simple and inspiring, the objective established was one of *NO REWORK*. This simple objective (although one very difficult to achieve) spawned many activities, all of which served to both increase the quality of the software delivered to customers and to increase the productivity of the organizations involved in developing software and executing the release process.

FORMAL SOFTWARE INSPECTIONS

Another way of viewing the No Rework objective is "do it right the first time". One of the most significant steps taken was to institutionalize formal inspections on all design documents and new or modified code. Prior to this time, the vast majority of code developed for the operating system was reviewed by one or more engineers. The key improvement was to use a formal process, and to require it for all changes.

A standard process was developed by a team of senior engineers, and all engineers in the organization were trained on how to perform a formal inspection. There was resistance that needed to be overcome in the organization, but with strong management support the program has been successfully implemented.

Full implementation of this program occurred too late to have a significant impact on Release 4.0, so Release 4.5 was the first release to benefit from this particular effort, with Release 4.7 to receive the full benefit. In order to address concerns raised by many of the engineers, we have chosen not to make visible detailed data about each inspection, even to the management team. Our objective was to improve our software quality, not assess our engineer's abilities. Some early data from the inspections, however, indicated that finding defects via inspections was substantially more productive (which really shouldn't be a surprise). Most code inspections involved three to five engineers, with both preparation and actual meeting inspection time. Our results indicated that, on average, only four hours of engineering effort was required to both find and repair each defect identified during formal inspections. This improvement by a factor of more 45x (compared to the 180 hours/defect with traditional testing just to find a defect) can be attributed to the fact that defects were found more easily, and that the cause of the defect was usually obvious at the time it was discovered.

ROOT CAUSE ANALYSIS

In addition to preventing defects with inspections, we initiated a program to better understand the actual cause of defects found during testing or at customer sites. The initial effort was aimed at doing a Root Cause Analysis of around 300 defects found during the testing of Release 3.0. Root Cause Analysis is a structured approach to determine the underlying cause of problems, which frequently requires digging down through "causes" that are really symptoms of another cause (the catch phrase is "ask why five times").

Our efforts involved categorizing the defects into how the defect was introduced (e.g. design problem, coding error), and then further analyzing the largest categories. A team of engineers, led by a manager, performed the analysis, and the effort resulted in 38 separate recommendations on how to prevent defects in the future. Over half of these could be implemented immediately (and were), and the remaining recommendations are being addressed over time.

One of the initial recommendations was to make this an ongoing process for all defects that are fixed. While we haven't been able to analyze 100% of the defects, we continue to analyze the majority of the defects, resulting in ongoing improvements to our development and inspection processes.

EMPHASIS ON PATCH REDUCTION

One of the most costliest forms of rework, and one that clearly has a negative impact on customers is the generation of patches for software that has already been released. As part of our rework objective, we focused efforts on reducing the need for patches on Release 4.0 and subsequent releases. Analysis of the patches we had produced for previous releases revealed that a substantial percentage of the patches were to correct defects that were known at the time the release achieved MR. While the exact cause of this was difficult to determine, the prevailing wisdom was that in many cases we did not correctly assess the potential impact of a defect.

On the basis of this information, a goal was established of no patches required for Release 4.0 on problems that were known at the MR of Release 4.0. The entire backlog of known problems on MPE/iX was reviewed, and a concerted effort made to repair all defects which could potentially justify the generation of a patch if encountered by a customer. In addition, as new problems were discovered in the testing of Release 4.0, a key consideration in determining whether or not to fix it prior to release was whether or not it would potentially need to be patched post-release.

It should be noted that this objective doesn't imply that patches won't be generated for Release 4.0. Unfortunately, there will be previously undiscovered problems which will require patches, and it is also unlikely that our assessment of the potential impact of defects was perfect. At the time this paper was written, we did not yet have sufficient data to determine how well we had met our objective, but we do know that we repaired many defects in Release 4.0 which we would have chosen not to correct in previous releases.

GENERAL DEFECT REPAIR DURING BACKEND

Once the backend process begins, there is in general a high sense of urgency to reach MR as soon as possible because revenue is produced only when new products begin to ship to customers. In order to reduce risk, it is desirable to minimize the amount of change to the contents of the release once the backend begins. Prior to Release 3.1, repairs to defects after the TC1 milestone were limited to critical problems found during the testing of the release, and the roll-in of patches available for previous releases. Only those changes approved by the team which daily monitors the problems discovered would be included in the release.

A policy change was made effective with Release 3.1 to give the development organizations the authority to include any additional defects they deemed appropriate after the TC1 milestone (except for the final build). The review team could still designate defects as "must fix" for the release, but they could not block the inclusion of additional defect repairs.

This change has somewhat increased the risk associated with the schedule of a release, since the additional defect repairs do occasionally introduce a new problem which must be addressed. The benefit of including additional defect repairs in a release is considered to far outweigh the risk however. Release 4.0 contained in excess of 350 defect repairs in the operating system software for problems reported by customer sites in prior releases, as well as repairs for over 750 defects in the networking products. We have estimated that roughly 200 fewer defects would have been fixed in Release 4.0 had we not made the policy change described above.

SOFTWARE RESILIENCY

As part of the overall effort to reduce rework, a project was initiated to improve software resiliency in the MPE/iX operating system. Software resiliency refers to the ability of the software to tolerate hardware and software exception conditions. A specific objective was to reduce the number of system aborts which could occur.

Stricter guidelines were developed regarding when the calling of the system abort procedure was appropriate, and wherever possible attempts were made to recover from the exception condition instead. For example, system aborts resulting from reaching table limits could in almost all cases be removed.

Some system aborts were removed from the code beginning with Release 4.0, and additional improvements will be made in each subsequent release. Combined with efforts to repair defects in the code which ultimately lead to system aborts, the overall reliability of the system will continue to improve over time.

NETWORK SOFTWARE TESTING

Based on an analysis done several years ago on the defects found in networking products, seven separate projects were launched to improve the overall quality of the networking software shipped to customers. This effort involved in excess of five engineering years of effort, and resulted in improvements to existing tests as well as the creation of additional networking related tests. In addition, more hardware systems were dedicated for testing networking software.

The areas addressed by the specific projects were abnormal termination testing for NS, NS and transport patch testing, startup and shutdown testing of the NS transport, NMMGR product and test enhancements, NS transport test improvements, IMF quality enhancements, and network system release testing improvements. Release 4.0 was the first release to benefit from this effort.

OTHER EFFORTS

In addition to the above specific efforts, a number of other activities are focused on improving the overall quality of HP 3000 software. These include:

- Improvements to the development environment used by software engineers to enable them to more effectively test the interaction of software modules during the development phase.
- Expansion of the VAB Prep program, which enables third party software suppliers for the HP 3000 to gain early access to pre-release versions of an MPE/iX release. In addition, both managers and software development engineers met with a number of Independent Software Vendors to discuss ideas on how to better work together to improve software quality.
- A test architect position was established within the system testing organization to provide leadership in improving the effectiveness and productivity of our testing process. A particular emphasis is to develop a strategy for more testing from a customer perspective.
- Continuous improvements are being made to the PowerPatch program, to both reduce cycle time and better include the defect repairs of most interest to customers. While this doesn't directly improve the quality of a release at the time of MR, it will allow customers who install the software several months after MR to easily install additional defect repairs, resulting in a higher level of quality than the original released version.

CUSTOMER FEEDBACK

It is always important for those of us in the factory to keep in mind that the ultimate determination of the quality of our software is the customer perception of quality. Over the years we have established many different criteria for determining when it is appropriate to begin shipping a release. Such metrics as number of hours of continuous operation without a failure, number of known defects per 10,000 lines of source code, incoming rate of new problems, etc. are all useful in making the judgment about when to release the software, but in reality these are just various approaches that attempt to predict how the software will behave in a customer environment. The customer who experiences a system abort in the first day of operation probably doesn't care that the factory ran for over 120 hours under heavy load with no failures. Similarly, if we release with less than 1 known defect per 10,000 lines of code, this won't really comfort a customer who is impacted by one of those defects, or by one we didn't discover.

Unfortunately, the science of testing a large and complex set of system level software is not exact. The bottom line is that human beings must make many judgment calls, using as much information as can be obtained. The other reality is that we cannot physically test this large set of complex software in every possible customer environment (if we want to ship the release in a finite period of time), nor can we always correctly predict the impact of a given problem on every possible environment or configuration. Since the ultimate judgment of our success lies with our customers, we have established several ways for obtaining feedback from customers about our software releases.

Prior to MR, we actively solicit subjective feedback from each test site, both internal and external to HP, regarding the overall quality of the release. We also include the response centers in this poll, since they are the first HP organization to be impacted if a release begins to ship prematurely. The release will not begin shipments until there is a strong consensus that the appropriate quality is present.

Once we begin shipments of a release, we pay very close attention to any escalated hot site situations, as well as patch requests. In particular we are looking for previously undetected problems which could impact a substantial percentage of customers.

More recently, we have established the practice of conducting a customer survey focused on the specific release. Our initial attempt at this type of survey was for Release 2.2, and the response rate to the survey was very encouraging. The survey was mailed to every customer who was shipped a copy of Release 2.2 as part of our software update service, and around 15% of the surveys were returned. In addition to gaining valuable information about how customers perceived the overall quality of the release, we also collected feedback on specific aspects of the system, such as the installation process.

We have also conducted a similar survey for Release 4.0, and we are still in the process of analyzing the results at the time this paper was written. The response rate has again been quite good (21%), and early indications are that the majority of customers rate the quality of Release 4.0 as being better than previous releases.

The survey questions were developed in conjunction with various software development teams, so that we could get feedback on how well we have met our development objectives, and on how to further improve the quality and usability of our products. In the Release 4.0 survey, which was mailed out to customers about 3-4 months after most customers received, but not necessarily installed, the release, we have sought information regarding the update process, the need for patch installation on Release 4.0, system interoperability, system management capabilities, and release documentation.

CLOSING COMMENTS

It is very important to Hewlett-Packard that we continue to increase the quality of the software on the HP 3000. In doing so, we make it easier for our customers to address the critical business needs for which they have purchased our products, which should in turn increase customer satisfaction with our products, and thus encourage repeat business with Hewlett-Packard. At the same, improving software quality, especially when done with techniques focused on "doing it right the first time", can significantly increase the productivity of the development organizations. These productivity improvements in turn allow a larger investment in functionality and products that address a wider range of customer needs, and thus increase business for Hewlett-Packard.

Given that efforts in improving software quality benefit both customers and HP significantly, it should not be surprising that a great deal of attention is devoted to maintaining and improving the quality of the software on the HP 3000.

PAPER NUMBER: 5006

TITLE: The MPE/iX Architected Interface Facility--A Case Study Overview

PRESENTER:

Lee Courtney
Monterey Software Group
10420 South De Anza Boulevard
Suite #C
Cupertino, CA 95-14
408-257-3781

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Past, Present and Future of Upgrading MPE/iX

Tad Olson, Hewlett-Packard Co.
David R. Smith, Hewlett-Packard Co.
Ruben Gallegos, Hewlett-Packard Co.

Installation has come a long way since the early days of MPE/V and Hewlett-Packard continues to invest in new technology to make it even better. This talk briefly covers the history of HP 3000 installation and focuses on the benefits of Hewlett-Packard's current and future installation strategy.

The AUTOINST installation tool has its roots in MPE/V and has evolved to meet the needs of customers on MPE/iX systems. The AUTOINST on MPE/iX 4.0 incorporates many enhancements and fixes directly based on customer feedback.

Hewlett-Packard strives to make the installation of the MPE/iX operating system on customers' machines as easy as possible. All the new 3000/9x7 systems come with the HP 3000 software preloaded. Hewlett-Packard plans to expand this program with the new 3000/992 systems, and with systems released in the future.

Hewlett-Packard is continuing to evolve the installation products and processes. Hewlett-Packard is actively pursuing the next generation of installation tools to meet HP 3000 customer needs. With the progression of the installation process and Hewlett-Packard's investment into future technology, the task of upgrading MPE/iX is becoming increasingly efficient and flexible. This presentation will inform HP 3000 customers of the benefits of the latest MPE/iX upgrade strategy from Hewlett-Packard.

Structured Query Language

The outside story

F. Alfredo Rego

Adager Corporation
Sun Valley, Idaho 83353-0030
U.S.A.

Telephone +1 (208) 726-9100 • Fax +1 (208) 726-8191
E-mail alfredo@adager.com

Copyright ©1993 by Adager Corporation

INTEREX paper 5008

NOW THAT HEWLETT-PACKARD has kindly produced IMAGE/SQL, we should do two things:

1. We should thank HP for the exceptional work that Jim Sartain's team did in integrating SQL (*Structured Query Language*, the industry-standard database language interface) into HP's award-winning IMAGE/3000 database management system.
2. We should understand SQL, so we can use it as a worthy complement to the other outstanding features that IMAGE/SQL provides.

What is SQL?
What is it not?

Thanking HP is the easy part. Understanding SQL is not as easy. And I do not mean learning the SQL syntax—I mean comprehending and grasping the nature, significance and meaning of SQL. In particular, understanding SQL means to be aware that *SQL is not a type of database management system by any means: SQL is simply an interface.* SQL is just a type of data sublanguage that some database management systems—such as IMAGE/SQL—happen to understand.

Why did I choose "***The outside story***" for my title instead of sheepishly following the standard cliché, "***The inside story***"? I selected the title very carefully, to emphasize the point that SQL is indeed an outsider. (There is nothing wrong with the fact that SQL is an outsider, as long as everyone is aware of this fact. Unfortunately, there is much confusion regarding SQL.)

In this essay, it is my objective to balance things out. After all, with IMAGE/SQL you now have the best of *both* worlds: you enjoy the *inside* strengths of IMAGE as well as the *outside* connectivity of SQL.

An Overview of HP IMAGE/SQL

Hewlett-Packard Company
Commercial Systems Division
Database Lab
19111 Pruneridge Avenue
Cupertino, California
Jim Sartain
(408) 447-5450

This paper represents the collected works and efforts of several individuals from the Database Lab and Marketing organizations of the Commercial Systems Division of Hewlett-Packard Company. This paper will be presented by any one of these individuals at every major Users' Group gathering in 1993.

****

Some of you are new to IMAGE, and others of you have been familiar with the database since the mid 1970's when it was winning awards for function and design. Thousands of applications, from email to finance to production control, all operate using IMAGE as their database management system. And if you count the users executing programs against data stored in IMAGE, their numbers would, no doubt, be in the millions.

Without argument, IMAGE gives you high reliability and high performance. After almost twenty years of improvements and refinements, you, as customers and users, would expect nothing less. With so much going for this product, with a dependability that few other database products offer, as a storage management tool that is a de facto standard on HP 3000 computer systems, what could possibly be done to improve it? The answer is quite simple. Standardize the access to IMAGE!

Hewlett-Packard is proud to introduce HP IMAGE/SQL, relational access to IMAGE data using industry-standard Structured Query Language (SQL). This method of access includes full read and write capability using ANSI standard functionality. This new access method makes a myriad of application development and decision support tools available to IMAGE that have never been available before.

The most astounding part of this whole story is that SQL access to IMAGE and the access that you enjoy today are completely compatible. Complete coexistence. No conversions, no recompilation, no changes. The only change will be the way you look at your data once you try the application development and decision support tools now available with IMAGE. And these are the same tools that can be used with all the big name relational databases, including ALLBASE/SQL. IMAGE has moved into the Open Systems arena.

Those of you familiar with IMAGE will undoubtedly be asking the question, "but what will this do to performance?" Of course, every database design and application will have it's own performance characteristics, but, as a general rule, IMAGE/SQL will perform at about seventy to ninety percent of native TurboIMAGE access. This term, 'native' refers to intrinsic level access using COBOL, PASCAL, et cetera. So, you will have the best of both worlds, fast native access and relational access.

With IMAGE/SQL you will have everything that you have today, wrapped in a new relational package, with an enormous selection of new tools, without any hassles. Let's take a look at the issues HP has tried to address.

TOP ISSUES

There are a multitude of issues which HP hopes to resolve with the introduction of IMAGE/SQL, and these will be explained shortly. Unfortunately, some new issues will be created as well, and these will also be revealed. These issues have been grouped according to the kind of organization that will be impacted, and they are VARs (Value Added Resellers) and ISVs (Independent Software Vendors), MIS or IT (Information Technology) Departments, and End-users.

Most VARs and ISVs, whose products include applications and tools, have been limited to non-relational application development tools when dealing with IMAGE. There are also very few PC-based GUI (Graphical User Interface) products which offer transparent interaction between PC clients and HP 3000 servers using IMAGE. As mentioned before, thousands of applications have been developed by uncountable companies all using IMAGE as the storage management system. Now, with IMAGE/SQL, all the new SQL-based application development and decision support tools, most using some type of GUI, can be used by VARs and ISVs to improve and enhance their products. This new transformation of IMAGE data into relational data will revitalize the VAR/ISV product environment.

Turning to MIS and IT departments, the primary issues are:
- o reducing the application backlog,
- o providing flexible information access,
- o recruiting SQL-trained personnel.

Each of these deserves some expansion.

Ideally, IMAGE/SQL should reduce the application development backlog for any MIS/IT organization. This reduction would be accomplished using any of the easy-to-use application development and decision support tools now available. Sadly, the opposite effect, that of increasing the demand for applications, will probably become true. As the end-users see the amazing results from new implementations using these tools, all kinds of previously hidden application requests will surface. Once the power of IMAGE/SQL is known, MIS/IT departments will be flooded with new requests for information.

Does this mean that these departments should avoid using IMAGE/SQL just to avoid this rampage? Absolutely not! The gains a company can realize from improved information review and analysis, as the result of improved data availability, can be significant. And MIS will not lose control. Users will be able to do their own queries and reporting, MIS need only supply the access. Some of the other issues in this area will further explain these statements.

The second issue in this area pertains to flexible information access. What this refers to is the powerful, command-driven language which characterizes SQL. Complex and sophisticated queries can be executed to supply end-users with their specific information requirements.

SQL uses command sentence constructs of SELECT and WHERE clauses, and the relational JOIN clause, to define its syntax. This level of knowledge is not necessarily required to use the various tools, however more sophisticated applications might require SQL expertise. Finding the right people to develop these queries and deliver these applications to the end-users will be critical. This brings us to the last issue for the MIS/IT department -- finding qualified personnel.

Relational concepts in information management have taken center stage in the last ten years. These concepts have become the de facto standard for data storage systems at educational institutions all over the world. As such, new and old computer professionals alike have had either some or extensive exposure to relational concepts.

SQL has become the relational language standard and is taught widely. Finding qualified personnel to program using SQL today is easier than finding experienced IMAGE programmers. So, even though your information investment is in IMAGE, the investment you make in accessing that information can be in relational technology.

The end-user has repeatedly come up in the last several paragraphs. Their issues are improved productivity and improved decision support. Both of these issues rest firmly on the same foundation; information that the end-user requires to be more productive and upon which decisions are made must be located where the user can exploit it and be presented in a meaningful form. This can be achieved with the use of PC-based client/server tools featuring graphical user interfaces (GUI). These are the very same tools the MIS/IT department will be using for development and that VARs and ISVs will use to enhance their offerings.

THE NEW TOOLS

The new tools have been mentioned over and over. Before getting to specific products, let's first describe them generally, and then break them into categories to better understand what they can do and offer.

All of these tools are PC-based and function primarily in a client/server environment operating with Microsoft Windows. With exception, these tools use ALLBASE/PC API, which is based on the Gupta standard SQL API, as the link to the server.

An Overview of HP IMAGE/SQL

For those of you not familiar with what an API is, some explanation is in order. API stands for Application Programming Interface and refers to the component of the client/server model which performs the interactive linking between client and server over the network for the purpose of data exchange. This can be thought of as application level handshaking, much the way RS232C is an electrical signal handshaking in data communications.

Besides Gupta, several other PC API standards exist. The use of the term 'standard' may seem vague as it is used here, but it basically refers to an agreed upon set of operating conditions. None of these standards represent an industry standard, but some type of API is required for client/server computing, so each vendor must select one or more standards with which to operate. So far, the Gupta standard serves the majority of the tools which have been certified for use with IMAGE/SQL. Microsoft has announced ODBC, its API standard, and chances are that this API will become a significant player in client/server technology and the basis for future certification, however, at the writing of this paper the product was not available. HP is planning availability of support for ODBC in late 1993.

Our first category of tool is Decision Support. As the name suggests, the purpose of these tools is to support business decisions, and this is accomplished through information analysis, reporting and graphical representation. Typically, this type of tool is oriented towards end-users doing financial and managerial analysis where numerical quantification and graphical representation of data is useful. These tools also tend to offer formatting of data for subsequent importing to spreadsheet products for further manipulation and review. End-user knowledge of SQL is not a prerequisite for using these tools.

Some examples of Decision Support Tools are Impromptu by Cognos, Quest by Gupta, and HP's own NewWave Access. Although this last tool is not new, the relational access now delivered through IMAGE/SQL will dramatically reduce the overhead which will improve administration of these systems and make this solution easier to setup and maintain.

Our next category of tools is called Executive Information System (EIS) Tools. The key aspect of these tools is their exception management capability, where information is monitored within user defined limits and deviations highlighted. These tools use colors and graphics extensively to bring attention to situations where limitations have been exceeded. Other features include trend analysis and information drill-down. Drill-down simply refers to digging out the detailed elements of summarized information. A good example of an EIS tool is Forest & Trees by Channel Computing which will be discussed in more detail later.

The last category of tools for use with IMAGE/SQL is Application Development Tools. Falling also into the category of Fourth Generation Languages (4GL), these tools offer PC Windows programming capability with Graphical User Interfaces which allow programmers fast development of critical end-user applications. In many cases, knowledge of SQL is not necessary.

Where Decision Support and EIS tools tend to be read intensive or read only, application development tools are intended to create interactive applications for online transaction processing. Some offerings in this category are PowerBuilder by Powersoft and SQLWindows by Gupta.

All the aforementioned categories make up the new tools available for use with IMAGE/SQL. These tools also operate with HP's ALLBASE/SQL and the other major independent relational databases. A review of some of these tools follows.

CURRENT TOOLS OFFERINGS

In addition to all the PC client/server tools mentioned above, a rich assortment of direct-access 4GL tools exist for accessing HP's relational databases. This prior sentence specifically indicates databases in the plural. All the tools in this section work with both ALLBASE/SQL and IMAGE/SQL. The important point about these tools is that they execute on the host, not the client/server cooperative execution of the previous set of tools.

The products in this section have, for the most part, been around for some time. Details will not be presented here about features and benefits. The main point here is that several vendors offer SQL-based tools for accessing relational databases.

Some of these 4GL tools offer native (intrinsic) access to IMAGE. However, the relational access they offer ALLBASE/SQL now extends to IMAGE/SQL. Below is a list of products and the companies which offer them:

ALLBASE Toolset	Hewlett-Packard
Transact	Hewlett-Packard
Powerhouse	Cognos
Focus	Information Builders
JAM	JYACC
Speedware	Speedware
Uniface	Uniface

As you can see, these tools, along with those indicated for PC client/server, make an impressive arsenal in your development efforts.

Two relational databases from HP can reside on your system, ALLBASE/SQL and IMAGE/SQL, and all these tools can access each. Most of what this paper describes is what IMAGE/SQL offers you. Some explanation of what ALLBASE/SQL offers follows.

WHEN TO USE HP ALLBASE/SQL

HP offers two database management systems on the HP 3000 platform. Each has its place with any given application, and many applications could use either. The majority of database usage on the HP 3000 is currently IMAGE, but there are certain applications where ALLBASE/SQL is the better choice.

ALLBASE/SQL is a full-featured relational database management system (RDBMS) with functional compliance of ANSI standards. HP sees four specific areas where ALLBASE is the preferred DBMS:

Mainframe class computing,
Distributed applications,
High-volume online transaction processing,
Object-oriented applications.

Below are expansions of these topics.

HP has positioned ALLBASE as the alternative to mainframe-class database management systems (DBMS). This works in conjunction with HP's mainframe downsizing strategy which offers high-end HP 3000 systems as Corporate Business Systems. Providing online backup and restructuring, and supporting very large file sizes, ALLBASE is the DBMS of choice in this area.

Distributed information and the applications which support them are also addressed within the ANSI standards for relational database systems. Here, again, ALLBASE has been specifically featured. Using two-phase commit protocols, distributed transactions can be ensured of completion and accuracy.

ALLBASE also supports Encina technology from TRANSARC Corporation. This technology provides a standardized method of transparent distributed transaction processing in a multi-platform environment. These distributed application features make ALLBASE the clear choice compared to IMAGE/SQL.

Where high-volume online transaction processing (OLTP) is the objective, and relational concepts are required, ALLBASE is again the best choice. This pertains mostly to new application development. Since ALLBASE has been designed from the ground up as a high-end relational database, applications requiring high-volume SQL OLTP will benefit.

Object-oriented computing is being seen more and more as a viable solution in many applications. Most assuredly, any application which has multimedia requirements can benefit from object technology. Here ALLBASE/SQL has the advantage. Object storage, especially that of video, photograph, audio, et cetera, requires data structures foreign to IMAGE/SQL. ALLBASE/SQL already provides the necessary storage with Binary Large Objects (BLOBs).

From these descriptions you can see that there are specific situations for ALLBASE/SQL.

COEXISTENCE

The HP 3000 now offers a complete range of data management choices. Native IMAGE still provides the highest performance in the industry for mission-critical OLTP business applications. IMAGE/SQL provides data access through the multitude of 4GL and PC client/server toolsets at a small performance premium. Client/server computing is clearly the emerging trend in information processing, and IMAGE/SQL is now an important element, ensuring the protection of your information investment.

ALLBASE/SQL provides the highest SQL performance of any relational database in the industry and provides support for distributed database and distributed transaction processing. With Corporate Business Systems, ALLBASE/SQL can handle the requirements of nearly any enterprise at a fraction of the cost of traditional mainframe solutions.

Together, these database management systems make a powerful team. And they work together. Concurrent access is possible by linking the two together within a single environment. Your applications, whether host-based or client/server, can simultaneously access ALLBASE/SQL and IMAGE/SQL information. That is coexistence at its best.

But there is more.

THE RELATIONAL PICTURE

HP offers you two relational databases, IMAGE/SQL and ALLBASE/SQL. But the story gets better. Also available on the HP 3000 are Oracle and Ingres, and each of these RDBMS products have their links to IMAGE as well, Oracle through their OracleConnect product, Ingres through their ALLBASE Gateway to IMAGE/SQL. This creates a SQL shell over every database management system available on the HP 3000. And with this shell comes all the 4GL and client/server tools which have been mentioned throughout this paper. This is a very powerful offering, indeed.

Taking a step farther, HP has proposed several object-oriented functions for addition to the SQL standard, the objective of which is the defining of Object SQL (OSQL). Should these proposals be accepted, object technology will be a step closer to IMAGE/SQL.

This last section gives a picture of where we are today and a glimpse of what the future might hold. Let's step back and review what has taken place in the last several years to get us where we are today.

HP DATABASE DEVELOPMENTS

By looking at just the past couple of years, it's easy to see that HP is actively working to make their database products superior. The year 1991 showed the following improvements:

IMAGE
- Dynamic Rollback
- DBChange Plus
- Multiple database transactions
- Multiprocessor tuning
- ALLBASE/TurboCONNECT

ALLBASE/SQL
- Third-party tools support
- Client/server APIs
- Interoperability with DB2
- Multiprocessor tuning

All of these enhancements improved the use and functionality of these products and helped scale the software into the new high-end multiprocessor hardware systems released in this timeframe. Where these 1991 enhancements addressed general functionality, the improvements released in 1992 offered specific features. These are:

IMAGE
- Generic and keyword searches
- Critical item update
- 4GB file sizes
- Corporate Business System tuning

ALLBASE/SQL
- Database shadowing
- Record level locking
- Remote unattended backup
- Stored procedures
- Business rules and triggers
- Two-phase commit (via XA interface)
- Additional third-party tools

An Overview of HP IMAGE/SQL

By far the biggest announcement, occurring late in 1992, was the introduction of IMAGE/SQL. Release of this product provided the following features to IMAGE:

IMAGE/SQL
- o Relational access via SQL
- o Stored procedures
- o Business rules and triggers
- o SQL PC API support

The year 1993 will not be without its share of enhancements, either. Announced for release in 1993 are:

IMAGE/SQL
- o Dynamic detail set expansion
- o DBINFO enhancements
- o Increased software resiliency
- o Performance tuning
- o ODBC support

ALLBASE/SQL
- o High-end performance tuning
- o TRANSARC Encina support
- o Increased I/O parallelism
- o High availability improvements
- o ODBC support

This impressive list of delivered and promised features is a clear demonstration of HP's dedication to providing the best data management systems on HP computer systems. But for all the features, if the product is not easy to use, the product will not be used. Let's next explore just how simple it is to use the new IMAGE/SQL.

ELEGANT SIMPLICITY

Accessing IMAGE relationally is as simple as 1-2-3.

Step 1: create the environment;
Step 2: attach IMAGE to the environment;
Step 3: access IMAGE relationally.

Each of these steps will be briefly described in this section.

Step 1: Create the SQL Database Environment

This step involves the creation of the SQL Database Environment (DBE). This structure contains control information about the data represented by it. This environment is created using the ISQL utility as follows:

```
:ISQL
ISQL => START DBE 'SQLDBE' NEW;
```

If a DBE already exists, this step can be omitted.

Step 2: Attach the IMAGE Database to the DBE

The attachment process examines the IMAGE database and places equivalent SQL-structure information in the DBE. This step can include a variety of mapping functions for fine tuning relational access. This step uses the IMAGESQL utility.

```
:IMAGESQL
> SET TURBODB CUSTDB
> SET SQLDBE SQLDBE
> ATTACH
```

The assumption in this case is that CUSTDB is an existing IMAGE database.

Step 3: Access IMAGE Data Relationally

Any number of methods could be employed in this step to demonstrate this access. In this case, ISQL is used.

```
:ISQL
ISQL => CONNECT TO 'SQLDBE';
ISQL => SELECT * FROM CUSTDB.CUSTOMERS;
```

The result of this query would be a tabled list of all entries in the CUSTOMERS data set. You can also display information about the database itself.

```
ISQL => SELECT NAME, OWNER FROM SYSTEM.TABLE;
```

This would return all table names associated with the SQLDBE environment.

Updating IMAGE databases is also very simple. Here is another ISQL example which updates a column (item) called PRODUCT_NAME.

```
:ISQL
ISQL => CONNECT TO 'SQLDBE';
ISQL => UPDATE CUSTDB.ORDERS
          SET PRODUCT_NAME = 'IMAGE/SQL'
          WHERE PRODUCT_NAME = 'IMAGE';
```

This update finds all entries where PRODUCT_NAME is 'IMAGE', then changes that value to 'IMAGE/SQL'.

As these steps demonstrate, accessing IMAGE data relationally is very simple and straight-forward. Once the database is attached to the DBE, nothing else is required in the regular use of the system. There are additional administrative tasks related to security and data type mapping, but none of these are overly complex or cumbersome.

Next, let's discuss the simplicity of some the client/server tools.

Forest & Trees by Channel Computing

If you recall, Forest & Trees (F&T) is a PC client/server tool falling under the category of EIS. This means this tool reads, reports and formats data from the IMAGE/SQL database (or any of the other host- or PC-based databases).

Quoting from the Reference Guide: "Forest & Trees... collects and combines data from a variety of sources and monitors the resulting information in order to track information at all levels from business vital signs to underlying detail.

"To help you display, integrate, and use the collected information, F&T has a large set of data manipulation and object control functions. These functions can be used in formulas, queries, triggers, graphs, and reports.

"Forest & Trees can also be fully customized with pictures, buttons, menus, and other graphic attributes to create distinctive applications."

Although the details of this tool are too extensive for this paper, some details will be discussed to demonstrate this tool's ease-of-use.

F&T has a tool bar at the top of the window which is filled with object icons for ease in application control and design. Views of data are defined by the user and extracted on demand or when scheduled. These views can be at any level, from high-level summary to low-level details. The drill-down feature of this tool allows display of all levels of information from top to bottom. Graphic views of the various levels of information can also be created and displayed.

One of the outstanding features of this tool is that user-defined limits or ranges can be set to monitor selected items, formula results, and sums. Color coded warnings, green for within limits, yellow for warnings, and red for outside limits, can highlight monitored data. Exception management is much simpler when the software isolates and highlights the exceptions.

PowerBuilder by Powersoft

Another PC client/server tool is PowerBuilder by Powersoft. This tool is from the Application Development category presented earlier and is both a reporting and transaction processing tool.

Quoting from a Powersoft sales brochure, "PowerBuilder... is a comprehensive Microsoft Windows-based development environment for constructing graphical client/server database applications through object-oriented development.

"PowerBuilder supports the rapid design and development of Windows-based client/server relational database applications for all marketplaces.

"PowerBuilder... contains full support for the MS Windows GUI technology, such as radio buttons, command buttons, list boxes, bit maps, etc., [and] supports custom user objects that can be defined as standard Windows controls."

As indicated, PowerBuilder is a very full featured application development tool. As mentioned above for Forest & Trees, this paper is not intended to be a detailed discussion of this products features. Instead, to summarize, this product offers significant productivity gains through development and use of its object technology orientation.

CONCLUSION

HP IMAGE/SQL is the most significant enhancement of the IMAGE DBMS of all time. The opportunity which this announcement provides to commercial and company developers is enormous considering the selection of tools now available for use with IMAGE. IMAGE/SQL also solidifies its coexistence with HP's other relational database, ALLBASE/SQL, with standard access to both.

The simplicity with which relational access occurs with IMAGE/SQL makes this perhaps the most painless 'conversion' in the history of the computer industry. The term 'coexistence' applies equally well to the dual accesses to IMAGE, native and SQL, as it does to the two relational databases, IMAGE/SQL and ALLBASE/SQL.

For all that this paper presents, the most important point is this: With IMAGE/SQL, HP has once again protected your investment in the HP 3000!

Paper #5011

A Programmer Looks at MPE/iX

by

Kevin Cooper

Hewlett-Packard Company
Commercial Systems Division
19055 Pruneridge Avenue
Cupertino, California 95014
(408) 447-4004

Presented at

The 1993 INTEREX Conference

San Francisco, California

September 19-23, 1993

Introduction

In 1987, Hewlett-Packard HP 3000 users began undergoing the most significant change in the history of the MPE operating system, when they began installing new machines that ran a distinctly different, yet upwardly compatible operating system known as MPE XL. To begin working with this new operating system, MPE users first had to install a new HP 3000 Series 900 computer. At that time, to help MPE programmers and system managers understand the important changes that came with MPE XL, this author published a paper called "A Programmer Looks at MPE XL."

In 1992, HP 3000 users began the second most significant transition in MPE's history, with the addition of POSIX/iX functionality. Again, MPE/iX has been designed to ensure upward compatibility from MPE XL. But this time, there is no need to install a new machine! MPE/iX runs on all the same hardware that MPE XL does. You enter the world of MPE/iX simply by updating your system to release 4.5 or later.

While a software upgrade may seem less significant than a box swap, HP 3000 system managers still need to plan carefully to minimize the possible impact to their users. Some individuals have expressed a reluctance to take this step forward, perhaps in part due to an incomplete understanding about exactly what has changed. So, this author took a cue from the film industry, and decided it was time to publish a sequel to the original paper, in order to clarify many of the issues HP 3000 users may face when they upgrade to MPE/iX.

Overview

This paper will adopt the perspective of MPE XL installed users who plan to upgrade their operating system to MPE/iX version 4.5 or later. Its goal is to help you become comfortable with the changes that have been introduced, so you can successfully plan for your system update. It will focus on the new features HP has included with the Fundamental Operating Software (FOS), which are available to all users of MPE/iX. Starting with release 4.5, FOS includes the facilities needed by end users to run software which was developed using the POSIX 1003.1 (known as POSIX.1) and 1003.2 (POSIX.2) standards.

Those users who wish to develop or port software using the POSIX.1 interfaces must acquire the MPE/iX POSIX Developers Kit, which is not included in FOS. This kit provides the C language compiler and header files needed to compile POSIX/iX source code, along with the POSIX.1 interface relocatable libraries (RLs) needed to link the compiled object modules into executable MPE/iX program files. The development or porting of POSIX/iX software is beyond the scope of this paper.

The first section of this presentation will describe the three most significant POSIX/iX additions to the MPE/iX Fundamental Operating Software: the hierarchical file system (HFS), a new file type called byte stream files, and the POSIX.2 shell and utilities. The second section will explore enhancements to familiar MPE XL functionality, mostly as viewed through the Command Interpreter (CI). The third section will outline the impact these changes have on MPE/iX system security and resource accounting.

Section 1 - New Features of MPE/iX

This section will describe three major new POSIX/iX capabilities of MPE/iX release 4.5. These are the hierarchical file system (HFS), byte stream files, and the POSIX.2 shell and utilities. These new POSIX/iX features have been integrated into MPE/iX at low levels of the operating system for maximum performance and reliability.

Hierarchical File System (HFS)

MPE files have followed the same naming convention since the beginning of MPE. File names contain 1-8 characters; the first must be a letter, while the others may be letters or digits. Letters are always shifted to upper case. File names may be qualified by group and account locations in the directory structure, with group and account names following the same naming convention as file names. In addition, there are several file domains where files can be stored, called permanent, temporary, and new. All of this continues to work exactly the same way on MPE/iX release 4.5.

The HFS adds multi-level directories to the permanent file domain. MPE account and group can be thought of as two levels in the directory structure - in fact, that is essentially how they are implemented in the new scheme. There is a new top layer called the root directory, specified by a slash ("/"). Users have the ability to add as many layers as they like below either the root directory or any MPE group. On release 4.5, HFS directories are not allowed at the account level; only MPE groups are supported there.

The HFS also adds a new file naming syntax called HFS syntax, used to represent HFS file and directory names. This new syntax is the default when you are running in the POSIX/iX environment, executing either the POSIX.2 shell or another program using the POSIX.1 interfaces. The rest of MPE/iX, except for a few utilities which have not been modified, uses a new hybrid syntax called "MPE-escaped" syntax, where the default is the familiar MPE file name. An MPE-escaped file name is interpreted using HFS syntax only when the first character of the file name is a dot (".") or a slash ("/"). The HPFOPEN and FOPEN intrinsics use MPE-escaped syntax, unless the programmer specifically requests HFS syntax or MPE-only syntax by setting a new item 41 in an HPFOPEN call.

Here are some examples of how the new MPE-escaped syntax works:

:PRINT myfile
prints the contents of MPE-named file "MYFILE" (upshifted)

:PRINT ./myfile
prints the contents of HFS-named file "myfile" (lower case)

When you use HFS syntax, upper and lower case letters represent unique characters, so "MYFILE", "MyFile", and "myfile" represent three different files. HFS names may also include three special characters, namely the dot (.), hyphen (-), and underscore (_). An HFS name may even begin with a dot or an underscore or a digit. HFS names can be up to 255 characters in length, except those residing directly under the root directory or in MPE groups, which are limited to 16 characters.

There are two ways to reference an HFS file name: absolute and relative. Absolute file names start with the root directory and list every directory traversed to reach the file. Thus absolute file names always begin with a slash. For example, the HFS file name "/rootfile" uses an absolute file name to describe the file "rootfile" directly under the root directory.

Relative file names use the new concept of the "current working directory" (CWD), which is introduced with the HFS. While this has some similarities to the familiar MPE concept of a logon group, there are also some important differences, which will be discussed later. You should think of your CWD as a short-hand for specifying file names, and no more. Relative file names use the CWD as a base and traverse directories from there. In HFS syntax, "myfile" without a leading slash refers to "CWD/myfile". MPE-escaped syntax treats unqualified MPE file names as relative to the CWD rather than the logon group, so "MYFILE" refers to "CWD/MYFILE". This will not impact users unless they explicitly execute a command to change their CWD.

Two special cases of relative file names are the dot (.) directory, which specifies the CWD, and the dot-dot (..) directory, which names the parent directory one level above the CWD. The meaning of the dot character in file or directory names depends on its context:

1. A separator between the MPE file, group, and account names.
2. A valid character in an HFS file or directory name.
3. The current working directory (when it appears alone).
4. The parent directory (when two dots appear together alone).

Since MPE groups and accounts are just special types of directories, all permanent MPE files can also be referenced with HFS syntax by specifying the name in the format "/ACCOUNT/GROUP/FILE" (upper case is mandatory). For example, "/SYS/PUB/EDITOR" is the HFS file name for the MPE-named file EDITOR.PUB.SYS. However, the converse is not true; HFS files outside MPE groups cannot be accessed using MPE file names. The file "/rootfile" can only be referenced using its HFS name.

On release 4.5, the :LISTFILE command was modified to handle the HFS, but the :LISTF command was not. :LISTFILE now defaults to listing the files in the CWD rather than in the logon group, but the :LISTF command still lists the files in the logon group, not the CWD. As part of the new HP-supplied User-Defined Command (UDC) file on release 4.5, called HPPXUDC.PUB.SYS, a UDC was provided for :LISTF, which calls :LISTFILE. Even when this UDC is set, the COMMAND intrinsic does not recognize UDCs, so it still uses the old :LISTF command.

With release 4.5, there are still several limitations on the types of files that can be used outside MPE groups. The most notable are executable files (NMPRG, PROG, XL, SL) and privileged files (PRIV), including all data base files. Other restricted files include Compatibility Mode (CM) files (KSAM, MSG, RIO, CIR), but not Native Mode (NM) KSAM files, which may be used outside MPE groups. Files in the NEW and TEMP domains, including system-defined files such as $OLDPASS and $NEWPASS, must reside in MPE groups. UDC files, user logs, system logs, system configuration files, and files used by the mirrored disk and SPU switchover software must also be in MPE groups.

Byte Stream Files

MPE/iX release 4.5 introduces a new type of file to the HP 3000, the byte stream file. Byte stream files are used extensively in the POSIX/iX environment. These files are similar to variable length record files without the concept of a record. Each file is simply a continuous stream of bytes, using the newline character ("\n" in C, which is a linefeed, or ASCII 10) to separate data. This significant new feature of MPE/iX is available to all users, whether using the HFS or not. A byte stream file can reside in either an HFS directory or an MPE group, with either an HFS file name or an MPE file name.

The MPE/iX file system treats byte stream files as variable length record files, so all commands and utilities which operate on variable length files will also work with byte stream files. This includes the :PRINT and :COPY commands, but not any of HP's editors (EDITOR, TDP, HPEDIT, HPSLATE). The :PRINT command, however, issues a warning when it begins printing a byte stream file, due to the possibility that more than 270 consecutive characters will be read before it reaches a newline character. This warning is issued before :PRINT knows whether any truncation will actually occur, since it has not yet read the file.

```
:PRINT bytestrm
Print output has been truncated to 270 characters. (CIWARN 9004)
```

This is the kind of thing that leads inquisitive minds to experiment. Here is a sample command file for :PRINT, which both eliminates this warning and tries to find the file using both MPE and HFS names. It does not handle all the options for the :PRINT command, but illustrates one way to handle these two situations for files in the permanent domain.

```
PARM filename
COMMENT   Save the message fence and turn off warning messages.
SETVAR _save_msg_fence hpmsgfence
SETVAR hpmsgfence 1
COMMENT   If the file exists as an MPE file, print it.
COMMENT   Otherwise, if the file exists as an HFS file, print it.
COMMENT   If no such file name exists, print an error message.
IF finfo("!filename",0) THEN
     ECHO :PRINT !filename
     PRINT !filename
ELSEIF finfo("./!filename",0) THEN
     ECHO :PRINT ./!filename
     PRINT ./!filename
ELSE
     ECHO Cannot :PRINT. Neither !filename nor ./!filename exists.
ENDIF
COMMENT   Reset message fence to its original value and clean up.
SETVAR hpmsgfence _save_msg_fence
DELETEVAR _save_msg_fence
```

The POSIX.2 Shell and Utilities

The new POSIX.2 shell and utilities represent another significant addition to the MPE/iX operating system. The .2 shell is an entirely new command interpreter on MPE/iX, which runs under the familiar CI. Its main purpose is to support the POSIX/iX application development and run-time environments. Users may choose either the CI, the shell, or some combination of both to accomplish various tasks on the system.

The POSIX.2 shell and utilities bring several new programming tools to the HP 3000, including new text editors (vi, ed, ex) and aids for searching (grep), revision control (rcs), lexical analysis (lex), parsing (yacc), data manipulation (awk), and program generation (make). The .2 shell and utilities currently only understand byte stream files, and cannot read or write MPE files with records. Two utilities have been provided to convert MPE record-structure files to and from byte stream files, called "tobyte" and "frombyte".

To start the shell from the CI, you can use the UDC "sh", included in HPPXUDC.PUB.SYS, or you can execute ":sh.hpbin.sys -L" directly if your CWD points to your MPE home group. This begins a new child process under the CI. The "-L" (upper case L) is required to initialize a "login" shell environment. Once in the shell, you can specify any of the new commands or utilities by name. Remember that the shell and its utilities use HFS syntax for all file names.

To execute an MPE/iX CI command from within the shell, you may use the "callci" shell command followed by the CI command with the appropriate parameters. If the CI command has more than one parameter, enclose the whole CI command in quotes so the shell does not try to interpret any semi-colons as the end of a shell command.

To terminate the shell process and return to the CI, enter the "exit" command. Pressing the BREAK key in the shell or its utilities will jump you out to the CI, not to the shell prompt. :RESUME will then return you to the place where you pressed BREAK. Most shell utilities can be interrupted by pressing Control-Y, which returns you to the shell, but there is no equivalent of a :RESUME command from that point.

You can execute shell commands directly from the CI by specifying the command name as a program file in HPBIN.SYS, which is where the shell and its utilities reside. You can even add HPBIN.SYS to your HPPATH variable, which will cause the CI to look for any unknown CI commands in the HPBIN.SYS group. For example, with the HPPATH variable set, "ls" entered at the CI prompt will execute the shell command "ls" to list files, and "pwd" will display your CWD. Some shell commands which require the shell environment will not work in this manner.

This paper cannot possibly discuss all the new capabilities available with the .2 shell software. For more information, refer to the HP two-volume manual set entitled "MPE/iX Shell and Utilities Reference Manual", which describes all the new shell functionality. You may also want to review the "MPE/iX Shell and Utilities User's Guide", which gives a more detailed look at many of the new utilities.

Section 2 - Supporting Enhancements to MPE XL

To integrate the POSIX/iX features into MPE/iX, many enhancements were made to the familiar MPE XL functionality. This section discusses how the changes introduced in release 4.5 will affect current MPE XL users.

Wild Card Characters in File Sets

Wild card characters used in MPE/iX file set parameters to CI commands like :LISTFILE and :STORE generally retain the same meaning they had before 4.5. However, there are some things to be aware of now that the POSIX/iX features have been added to MPE/iX. These mostly involve the use of the "@", but there are also changes with the hyphen ("-").

It turns out that the meaning of the "@" in a file set depends on the context in which it is used. The shell uses the asterisk ("*") as a wild card character, and this is not interchangeable with the "@" used by the CI. Here are some examples of what "@" means:

```
:LISTFILE @    { lists all files in the CWD with valid MPE names }

MYFILE      YOURFILE

:LISTFILE ./@
     { lists all files in the CWD with valid HFS names }
     { note that upper case names sort before lower case names }

 PATH= /COOPER/PUB/./

 MYFILE              YOURFILE            long_file_name    mydir/

:LISTFILE ./@;TREE    { similar to :LISTFILE ./ }
     { recursively traverses directories under the CWD }

 PATH= /COOPER/PUB/./

 MYFILE              YOURFILE            long_file_name    mydir/

 PATH= /COOPER/PUB/./mydir/

 FILE2  file1
```

When "@" is used in MPE syntax, it means all valid MPE-named files. When it is used in HFS syntax, it means all objects, including MPE files, HFS files and directories. This has significant implications for :STORE and :RESTORE, which will be described in the next topic.

Since hyphen ("-") may now be a valid character in an HFS file name, the range [a-z] could have meant either [a through z] or [a or - or z] in HFS file sets. The first meaning was retained for compatibility. The way you specify the second option is to put the hyphen next to one of the brackets [-az] or [az-]. The ranges [A-Z] and [a-z] mean the same thing in an MPE file set, but the HFS is case sensitive and will only match the case you specify.

:STORE and :RESTORE

The meaning of "@" discussed above has a major impact on backing up your system. On release 4.5, when you execute ":STORE @.@.@", ":STORE @.@[.ACCOUNT]", or ":STORE @[.GROUP[.ACCOUNT]]", you only store files with valid MPE file names which reside in an MPE group. You do NOT back up any files that can only be referenced with HFS syntax. MPE/iX issues warnings when this occurs, but system managers must be aware of the need to change backup procedures on release 4.5. To do a full backup on a 4.5 system, you must specify "**:STORE /**". To store an entire account, such as SYS, including all the HFS files and directories within SYS, you must specify ":STORE /SYS/".

If you do not modify any backup jobstreams and command files which use ":STORE @.@.@", you will not back up HFS files such as "/rootfile" (directly under root), "./this_is_my_file" (lower case, long file name, or special characters), or "./dir1/file1" (not in an MPE group). Every system running 4.5 FOS has some HFS files in directories directly under root, placed there by AUTOINST during the update process. These files and any HFS files created by users will not be backed up on 4.5 unless the ":STORE @.@.@" commands are changed to ":STORE /".

A similar change is required for restoring files. When you specify ":RESTORE ;@.@.@" from a tape containing HFS files, you will not restore files which can only be named with HFS syntax. To restore all files from a tape containing HFS files, you must specify "**:RESTORE ;/**".

Even though MPE XL :RESTORE versions prior to 4.5 (and all versions of MPE V) do not understand HFS file names or directories, you can :RESTORE an HFS file from a 4.5 :STORE tape onto a system running an earlier release of MPE XL. Every 4.5 :STORE tape with HFS files on it also contains a special file called HFSMAP, which maps each HFS file on the tape to a unique MPE file name, starting with "F0000000" and increasing sequentially. These HFS files and the HFSMAP file are placed in a special group and account, both of whose names begin with an underscore. This prevents you from directly restoring them onto a pre-4.5 MPE XL system, but you may :RESTORE them by using the ;LOCAL option or specifying the ;GROUP= and ;ACCOUNT= options. You can then determine each file's original HFS name by looking at the HFSMAP file.

While byte stream files can be restored onto a pre-4.5 system, they cannot be accessed by any MPE XL operating system version before 4.5. This is also true for HFS directory structures, which are copied to a 4.5 :STORE tape unless you specify the new ;TRANSPORT=MPEXL option. If you do :RESTORE one of these types of files onto a pre-4.5 system, you cannot get rid of it with the :PURGE command. You must execute a :PURGEGROUP or :PURGEACCT command to remove it from the system.

One other subtlety with :STORE and :RESTORE is that a blank space is required before the hyphen when a negative file set appears after an HFS file set (such as "/ -@.PUB.SYS"). This prevents any ambiguity, since the hyphen could otherwise be treated as part of a valid HFS file name. With MPE file sets, this trailing blank space remains optional.

New CI Commands and Variables

To manage HFS directories, four new MPE/iX CI commands and one new CI system variable were introduced on release 4.5. Each of these has a similar command in the .2 shell, which may be familiar to users of other operating systems. While HFS directories will mainly be managed in the POSIX/iX environment by using shell commands, corresponding CI commands have been provided to make the system manager's job easier.

The new commands, and their related shell commands, are:

:NEWDIR	mkdir	to create a new directory
:CHDIR	cd	to change your Current Working Directory
:PURGEDIR	rmdir	to remove a directory
:DISKUSE	du	to show disk space used in HFS directories

The first parameter to all of these new commands is the directory name, but the CI and shell use different naming syntaxes. The CI expects the directory name to be in MPE-escaped syntax, for compatibility with other MPE/iX commands, and will upshift the name if it does not begin with a dot or a slash. The shell expects the directory name to be in HFS syntax, and does not upshift it. Therefore, the MPE/iX CI command ":NEWDIR ./dir1" and the shell command "mkdir dir1" both create a new directory "dir1", but ":NEWDIR dir1" gives a different result because it upshifts the newly-created directory name to "DIR1".

Creating an HFS directory with :NEWDIR or "mkdir" is similar in many ways to creating an MPE group. The new directory becomes a repository for files (and other directories). The new ;CREATE=PATH option on the :RESTORE command also creates directories, when it is necessary to give a new disk file the same HFS file name it has on tape, assuming the user has the appropriate privileges to do so.

Switching your CWD with :CHDIR or the "cd" shell command is quite different from changing MPE groups. Your file access privileges do not change like they do when you execute the :CHGROUP command. :CHDIR never requires a password like :CHGROUP may, since it does not give the user access to any new files or capabilities. As stated earlier, the CWD should be thought of simply as a short-hand way for naming files, so changing it only impacts which directory you examine when you specify a relative file name in any of MPE/iX's file naming syntaxes.

The new system variable is called HPCWD, which stores the value of your CWD. It is similar to the shell environment variable PWD. It is set by the operating system whenever you change directories, but it cannot be set directly by the user, since it is a read-only variable.

Directories are removed with the :PURGEDIR command, similar to the shell "rmdir" command. :PURGEDIR can be used to remove directories recursively by specifying the ;TREE option or by ending an HFS syntax directory name with a slash. Any files in the directories affected will be purged as well, the same as when you purge an MPE group. You have the chance to confirm the purge before it takes place, unless you specify the ;NOCONFIRM option or execute the :PURGEDIR from a job.

The :DISKUSE command is described under "System Resource Accounting."

CI Error Handling

To accommodate the HFS, some CI error handling has been changed. In almost every case, error numbers have not changed for existing errors, but new errors have been added where required to handle new situations. Let's look at an example which is fairly typical, a :LISTFILE where there are no files in the group. On pre-4.5 releases, users see:

```
:LISTF
NO FILES FOUND IN FILE-SET (CIWARN 431)
:LISTFILE
NON-EXISTENT FILE @.PUB.COOPER. (CIWARN 920)
```

On 4.5, for these same situations, users now see:

```
:LISTF { using the new UDC which calls :LISTFILE }
Non-existent MPE named file "@.PUB.COOPER". (CIWARN 920)
No MPE-named files found in file set. (CIWARN 431)
:LISTFILE
Non-existent MPE named file "@.PUB.COOPER". (CIWARN 920)
```

But new messages have been added for new situations, such as:

```
:LISTFILE ./@ { with no files which match }
No match found for the pathname. (CIWARN 9043)
:LISTFILE /SYS/NOGROUP/@ { where the group does not exist }
A component of the pathname does not exist. (CIWARN 9053)
:LISTFILE /SYS/PUB/EDITOR/@
A component of the pathname is not a directory. (CIWARN 9055)
```

You may have noticed that these new CI error messages use both upper and lower case, unlike the old messages, which used only upper case. At the same time these new messages were added, the existing CI error messages were edited to use both upper and lower case for readability. But the messages themselves have changed very little, except where it was necessary for clarification, such as the addition of the words "MPE named" to the :LISTF and :LISTFILE messages above.

An unusual case where the error number has changed is when you specify a non-existent MPE source file in the :COPY command (such as NOFILE in the examples below). On releases prior to 4.5, users would see:

```
:COPY NOFILE,NEWFILE
NON-EXISTENT FILE.  (CIERR 907)
```

On 4.5, in the same situation, users see:

```
:COPY NOFILE,NEWFILE
The last component of the pathname "NOFILE.PUB.COOPER" does not
exist. (CIERR 935)
```

To those who are not used to working with the HFS, the "pathname" referred to in this message is the filename expressed in HFS syntax, /COOPER/PUB/NOFILE. The "last component" of the pathname is then the filename itself, NOFILE.

This is one of the rare cases where the actual error number, returned in the CI error message and the HPCIERR variable, has changed. You will want to take note of this if you have jobstreams, command files or programs which check for this specific CI error number.

Special Cases

While we're discussing :COPY, there is one special situation worth mentioning. :COPY has always allowed you to copy a file to a different group by specifying ":COPY OLDFILE,.NEWGROUP". Since the dot character can now begin a valid HFS file name, this could be ambiguous, with ".NEWGROUP" meaning either an HFS file name or an MPE group. For this special case, the dot retains its previous meaning, and the command will work as it always has. To copy "OLDFILE" to a new HFS file named ".NEWGROUP", you must specify ":COPY OLDFILE,./.NEWGROUP".

Certain MPE/iX commands, such as :FCOPY, :STREAM, :XEQ (implied :RUN), and :WELCOME, still require MPE-only syntax for the file names given to them as parameters. One way to work around this is to use :FILE equations to refer to HFS file names, since :FILE equations allow HFS names on the right side of the equation. For example, to execute a command file with a name that can only be specified using HFS syntax, you could do:

```
:FILE mycmd=./my_command_file
:XEQ *mycmd
```

There does not appear to be a way, however, to take advantage of the HPPATH variable using this method, as the HPPATH variable has not been modified to accept HFS directory names.

The first few times you change your CWD to an HFS directory on MPE/iX (using the :CHDIR command), you may experience some surprises when you use certain MPE/iX features. An example is trying to build a file in the temporary domain with an unqualified file name, which will be treated as CWD-relative. This causes an error, because TEMP files cannot reside in HFS directories.

```
:CHDIR ./dir1
:BUILD WORKFILE;TEMP
INVALID FILE REFERENCE  (FSERR 54)
Build of file "./WORKFILE" failed. (CIERR 279)
```

You need to qualify the file name to override the CWD default:

```
:BUILD WORKFILE.!HPGROUP;TEMP
```

A similar problem occurs when you try to create files in the NEW domain while the CWD does not point to an MPE group. For example, trying to close the "TO" file as a NEW file with FCOPY results in an error:

```
:CHDIR ./dir2
:FCOPY FROM=OLDFILE;TO=NEWFILE;NEW
INVALID FILE REFERENCE  (FSERR 54)
```

Since all HP editors create their files in the NEW domain, they will not run properly unless the CWD points to an MPE group.

The MPE/iX system-defined files $OLDPASS and $NEWPASS are handled specially. They always default to temporary files in the logon group, and are not treated as CWD-relative. HFS syntax does not allow you to specify file names beginning with "$", such as these, and therefore you also cannot specify "./$" in MPE-escaped syntax. The other MPE/iX system-defined files ($NULL, $STDIN, $STDINX, $STDLIST) are also independent of where the CWD is located, and can be accessed using MPE syntax only.

As you can probably imagine, the new POSIX/iX features have a definite impact on system security. POSIX security requirements have been integrated into the MPE security system so that both work together. Whichever type of security features you use, MPE/iX protects your system and its data files from unauthorized access. This section explores how some of the new POSIX/iX security functions have been implemented and how they affect existing MPE security. It concludes with a discussion about changes to system resource accounting.

Integrating POSIX/iX Security

Logon security remains basically the same. Everyone logs on as a USER.ACCOUNT, with the MPE group specified explicitly or allowed to default to the user's home group. Two new concepts are POSIX User ID and Group ID, which you should be aware of even if you are not using POSIX/iX. As of release 4.5, every MPE user/account combination must have an entry in the POSIX User ID file, and every account must have an entry in the POSIX Group ID file. A new MPE/iX utility called PXUTIL automatically runs when you update your system to release 4.5, to create these two new files. If these files somehow get "out of sync" with your directory, only users with System Manager (SM) capability will be able to log on. Other users will see the message:

```
     Out of System Resource(s). (CIERR 629)
```

To correct this situation, you should log on as MANAGER.SYS and run PXUTIL, then select the UPDATE command to re-synchronize these two files with your directory.

The changes to file security are much more significant, but mostly affect files outside MPE groups. Security for files in MPE groups is still mainly based on the location of the file in the system directory. To access a file, a user needs permission to access the account, the group, and the file. If the file has a lockword, the system requires that to be specified as well. The exception to these requirements is when the file has an Access Control Definition (ACD) associated with it, but this feature has not been widely used.

POSIX/iX bases its file security on the concept of the ACD, so this capability will see much more use on release 4.5 and later. ACDs specify exactly which users may access a file. They can be much more selective than the MPE file access matrix, where the only way to allow even one user outside the file's account to access the file was to allow access to ANY users. To meet the POSIX security standards, all files which are outside MPE groups and all HFS directories (but not MPE accounts and groups) now have ACDs associated with them.

HFS directories use five new security modes in their ACDs, which may be used to restrict access to a directory:

TD - Traverse Directory entries
CD - Create Directory entries
DD - Delete Directory entries
RD - Read Directory entries
RACD - Read the ACD for this directory (also used for files)

Some of the ways security checking is enforced for directories are:

:CHDIR	needs TD to all HFS directories in the new path
:BUILD	needs CD to the new file's parent directory
:PURGE	needs DD to the file's parent directory
:NEWDIR	needs CD to the new directory's parent directory
:PURGEDIR	needs DD to the directory's parent directory
:LISTFILE	needs TD to all HFS directories in the path and RD to the directory having its objects listed
:LISTFILE,-2	needs RACD to the directory having its ACD listed

Since MPE began, the :LISTF command and its successor, :LISTFILE, have allowed you to list files anywhere on the system. This is still true on release 4.5 for files in MPE groups. But new rules apply to objects in HFS directories, to comply with POSIX security standards. To list the contents of an HFS directory, POSIX requires a user without an appropriate privilege (such as SM capability) to have TD access to all directories in the path being searched and RD access to the desired directory. So, to list the files in directory "/dir1", a user without SM capability must have TD access to the directories root ("/") and "dir1", and RD access to "dir1".

The internal format of the ACD has changed with release 4.5. This can cause a problem if you :RESTORE a file with an ACD from a 4.5 :STORE tape onto a system running an earlier release of MPE XL. The new ;TRANSPORT=MPEXL option on the 4.5 :STORE command avoids this problem by placing ACDs on the tape in the pre-4.5 MPE XL format. The existing ;TRANSPORT option, which moves files to MPE V systems, does not copy ACDs to tape unless the ;COPYACD option is specified in the :STORE command. In that case, the ACDs are translated to MPE V format ACDs.

If a 4.5 :STORE tape was created without using the ;TRANSPORT=MPEXL option, you can still :RESTORE its files onto a system running a pre-4.5 MPE XL release by specifying the ;NOACD option on the :RESTORE. You should reapply the ACD with :ALTSEC after the :RESTORE completes.

If neither the ":STORE ;TRANSPORT=MPEXL" nor the ":RESTORE ;NOACD" option is used, and you :RESTORE a file with a 4.5 ACD onto a system running a pre-4.5 MPE XL release, you may see some errors later. Only users with SM or AM (Account Manager) capability, or the file creator, will be able to access this file. Those users will not notice any problems unless they try to access the ACD for the file with a command like ":LISTFILE,-2", when the following message will be displayed:

```
ERROR ENCOUNTERED WHILE ACCESSING ACD
```

All other users will not be able to access the file at all, and will see error messages such as FSERR 140:

```
ACCESS DENIED DUE TO CORRUPT ACD, FILE OWNER MUST REAPPLY ACD.
```

Changes to Long-time MPE Security Restrictions

Several long-time security restrictions in MPE have been changed, to allow POSIX security standards to co-exist with MPE/iX security. It is important to realize that these changes apply whether you are using any POSIX/iX features or not. If you have programs, jobstreams, or command files that depend on some of these restrictions, you should test them before you go into production on release 4.5 or later.

Saving Files Outside the Logon Account

A user with the appropriate capabilities can now save a file across account boundaries. This is provided because of a POSIX requirement, but it is true whether you are using any POSIX/iX features or not. A file in one account might therefore have a creator in another account. Or, from a different perspective, a user from another account might be the creator of a file in your account! The creator field in the file label held only eight bytes for the user name before 4.5, but now has room for 16 bytes, to hold both the user and account names.

Before 4.5, a user logged on as MANAGER.SYS, with SM capability, saw:

```
     :BUILD SMFILE.PUB.COOPER
     SECURITY VIOLATION  (FSERR 93)
     BUILD OF FILE SMFILE.PUB.COOPER FAILED.  (CIERR 279)
```

With 4.5, this same command will execute without error, and the resulting file will have "MANAGER.SYS" in the creator field. The file will not have an ACD assigned in this situation, but will use the standard MPE file security matrix for PUB.COOPER.

Since you can save across accounts, you can also :RENAME files across accounts. Before 4.5, you could not do so, even as MANAGER.SYS:

```
     :BUILD SMFILE   {in PUB.SYS}
     :RENAME SMFILE,SMFILE.PUB.COOPER
     SECURITY VIOLATION  (FSERR 93)
     RENAME FAILED DUE TO FILE SYSTEM ERROR, NOT RENAMED.(CIERR 373)
```

On 4.5, this :RENAME command works for an SM user. After the :RENAME, the file creator will still be MANAGER.SYS. In this case, the file is assigned an ACD during the :RENAME. The new ACD reflects the file security attributes of PUB.SYS, where the file was originally created.

Another case where a user can save a file outside the logon account is when the ACD for an HFS directory allows Create Directory (CD) access to ALL users on the system ("@.@"). This is how MPE/iX implements the POSIX specification for granting "write" (w) access to users of class "other". Any user on the system can save a file into such a directory. This can only be done with HFS directories, not MPE groups.

One situation where this may come up on any system is the special directory "/tmp", which is provided to implement the concept of temporary files on a POSIX system. This should not be confused with MPE/iX's temporary file domain, which is a totally different concept. By convention, files in /tmp are expected to exist for only a short time, and could be purged at any time. Since the default ACD for the directory /tmp allows CD access to "@.@", any user on the system (even with no special capabilities) can save a file there, by specifying:

```
     :BUILD /tmp/kcfile
```

Since the ACD for /tmp also allows Delete Directory (DD) access to "@.@", any user who creates a file there can also purge the file. Other users must have SM capability to do so, since the ACD assigned to the file itself at its creation restricts their access to it.

Renaming Files When not the Creator

One of the most requested enhancements to MPE over the years has been to remove the restriction that allowed only the file creator to :RENAME a file. Even if a user could validly copy the file and then purge it, MPE from the beginning has not allowed even the system manager to :RENAME a file without being its creator. Users would see:

```
:RENAME YOURFILE,MYFILE
USER IS NOT CREATOR  (FSERR 94)
RENAME FAILED DUE TO FILE SYSTEM ERROR, NOT RENAMED.(CIERR 373)
```

With release 4.5, there are now several situations where you no longer need to be the file creator to :RENAME a file. In every case, the user must have Save Files (SF) capability, and the file must not have a lockword. A user with SM capability can then :RENAME it to anywhere on the system, even if that user is not the creator. Other users may do so if they have TD access to all the directories involved, DD access to the file's current parent directory, and CD access to the proposed new directory. To maintain system security, an ACD reflecting the original MPE group security is automatically assigned to any file without an ACD, when that file is renamed from an MPE group to a directory that is not an MPE group in the same account.

This is where understanding all the security ramifications of mixing the MPE and HFS environments can get quite complicated, but it does all work. For more details, refer to the "MPE/iX Reference Supplement" sections for the :RENAME command and the FRENAME intrinsic. It remains true that only the file creator can turn file security checking off and on using the :RELEASE and :SECURE commands.

Managing Files with Lockwords

There are a few changes to watch for involving file lockwords. First, lockwords will only be enforced for MPE-named files which are located in MPE groups and have no ACDs. This is because security checking for a file with an ACD ignores any file lockword, and all files outside MPE groups have ACDs associated with them.

Also, lockwords may only be specified when using MPE syntax. Without this restriction, what would QUERY/LOCK represent in HFS syntax? It could be either the file QUERY with a lockword LOCK, or the file LOCK in the directory QUERY.

The other change to lockwords involves an error condition that has been removed. Before 4.5, if you specified a lockword for a file that did not have one, it caused an error:

```
:BUILD MYFILE
:PURGE MYFILE/LOCK
LOCKWORD VIOLATION  (FSERR 92)
UNABLE TO PURGE FILE MYFILE.PUB.COOPER. (CIERR 384)
```

On 4.5, this no longer causes an error, and the extraneous lockword is ignored.

Even though the restriction that allowed only the file creator to :RENAME a file has been removed, it is still true that only the creator can assign a lockword to a file. So, if you are not the creator of the file YOURFILE, you will still see the following error:

```
     :RENAME YOURFILE,YOURFILE/LOCK
     USER IS NOT CREATOR  (FSERR 94)
     RENAME failed due to file system error.  Not renamed. (CIERR 373)
```

System Resource Accounting

How does the system handle resource accounting for the HFS? First, remember that all users still log on to a group within an account. All CPU and connect time resources consumed by these users accumulate to the logon group and account, just as they always have. Changing directories has no impact on this; all resources still accumulate to the logon group.

The one new change we must consider has to do with accounting for disk space. The disk space limits and disk space usage for directories created under an MPE group are tracked as part of the group, so this presents no problem. But directories can also be created directly under the root directory, and these do not fall under any MPE group. System managers should be aware that disk space can be allocated in these directories without accumulating to the limit of any account or group, and without appearing in the :REPORT command totals.

The way to examine this use of disk space is with new :DISKUSE command. One way to invoke :DISKUSE is to specify ":DISKUSE /@", to show the disk space in use for all directories directly under root, including all MPE accounts. ":DISKUSE /" recursively shows all disk space in use by all directories, and lists the disk space used by all files directly under the root directory ("/") just before the end of the listing, on the line that displays the count for "(files directly below specified directory)". ":DISKUSE /;NOTREE" will show you the total for the whole system without all the detailed line items. The root directory ("/") can be replaced by any other directory in these commands to view just the disk usage for that specific directory.

The default system security settings limit the spread of disk use outside MPE accounts. Only users with appropriate privilege (SM capability) may create files or directories under the root directory and all but one of its child directories, since the ACDs for these directories do not grant CD access to any users on the system.

The one exception is the directory "/tmp", where POSIX/iX programs expect any user to be able to create a file. System managers need to establish a way to clean this directory out on a regular basis. One fairly safe method to accomplish this would be with the command:

```
     :STORE /tmp/;;DATE<={some recent date};SHOW=OFFLINE;PURGE
```

Another way is to execute the following commands occasionally:

```
     :CHDIR /tmp
     :PURGEDIR /tmp/;NOCONFIRM
```

This will remove the files in /tmp, but leave the directory structure in place, similar to what happens when you execute a :PURGEGROUP while logged in to the same MPE group you are purging.

Conclusion

There is no way anyone could possibly cover all the new features of MPE/iX in one presentation. But hopefully you have learned enough about MPE/iX so that you feel comfortable with its new functionality. You should now be able to identify the key issues of updating to release 4.5 and understand how they apply at your site. MPE/iX release 4.7 will introduce many more enhancements to the POSIX/iX features, including improvements for some of the MPE/iX integration issues described in this paper. So, whether you are planning to update to release 4.5 or wait for 4.7, you should now be much better prepared to plan for your update. Soon you will be taking advantage of many of these new features of MPE/iX, and looking back on this transition in the same way you now look back on your move from MPE V to MPE XL.

About the Author

Kevin Cooper has worked for Hewlett-Packard with the HP 3000 computer system since 1976, in software development, support and technical consulting. He currently assists HP 3000 software vendors with POSIX implementations, as part of the Channel Partners Consulting group at Commercial Systems Division. He holds a bachelor's degree in Computer Science from the University of California, Berkeley. He has been a presenter at three previous INTEREX conferences.

PAPER NUMBER: 5012

TITLE: How Did Image Become Relational?

PRESENTER:

Brad Tashenberg
Bradmark Technologies
4265 San Felipe, Suite #800
Houston, TX 77027
713-621-2828

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Paper #5013

Optimizing the Usability of HP OpenView System Manager: A Usability Engineering Case Study

Christine N. Gandel

Hewlett-Packard Company
Commercial Systems Division
19447 Pruneridge Avenue 47UC
Cupertino, CA 95014 USA
(408)447-7952

HP OpenView System Manager and HP OpenView Console

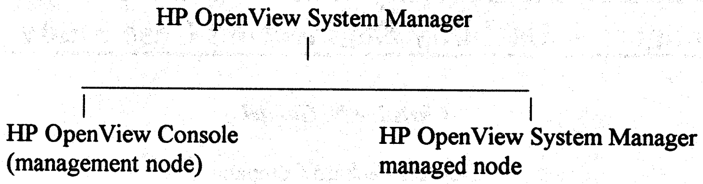
The goal of the HP OpenView System Manager product is to reduce the cost of managing an HP 3000 computing environment through increased operator efficiency and higher system availability. The product provides several key benefits:

- centralized monitoring and control of the environment
- management by exception
- task-based consoles
- scripted responses to system events
- enhanced message logging and tracking

HP OpenView System Manager is PC-based. It has been available for the HP 3000 platform for some time. However, at the time of the MPE/iX 4.0 and HP 3000 Corporate Business Systems release, there was a major enhancement to HP OpenView System Manager, the HP OpenView Console product. The HP OpenView Console product is an optional replacement for the terminal-based system console that provides, in addition to the standard console functionality, the benefits listed above.

Because of the critical nature of the functionality provided by this product, we felt it was important to expend extra effort to ensure that the product satisfied the needs of the users in their expected environments as fully as possible.

The structure of the product evolved during the period of time in which these studies were performed. I will therefore spend just a minute explaining the current product structure and the product(s) that were the subject of our usability engineering studies. The current structure of the product is as follows:



The initial testing about which I will report in this paper focused on the HP OpenView Console product, the single system or management node portion of HP OpenView System Manager. The later testing simulated a networked environment in which there were several systems being managed by the full HP OpenView System Manager product. I will use HP OpenView Console (OVC) to refer to the management node only product and HP OpenView System Manager (OVSM) to refer to the networked product. Note that OVC is a required portion of OVSM.

Definition of Product Usability

There are two major components of product usability, namely the product's functionality and the accessibility of the product to the user. A usable product has the right functionality and that functionality is easily accessible by the user. What does this mean?

It means that we are shifting our emphasis from the point of view of the developer of the software or system (what features should be implemented and how) to the point of view of the user (what actions need to be performed). Thecode development engineer has been trained in the former, while the Human-Computer Interaction (HCI) or Human Factors (HFE) engineer has also been trained in the latter.

It means that the usable product is:

- effective in supporting the user's task(s)
- easy to learn and use
- productive and efficient to use
- comfortable and satisfying to use

Thus the user can easily learn to use the product and is able to efficiently perform their tasks in their environment. They are satisfied by the procedures they must follow and the resulting output. They are effectively protected from consequences of their actions in that the system's integrity or the user's health and safety are not compromised when they make an error.

A usability defect is anything that makes it difficult or unpleasant for the user to accomplish their tasks in their environment. A usability defect may refer to an omission of functionality as well as to existing functionality. We measure usability defects in much the same way as we measure defects in reliability of a system, using similar criteria.

There are now a large number of processes that are commonly employed to ensure the usability of computer systems and software. Ideally, usability evaluation processes are applied iteratively during the product development life cycle. Good design does not imply that you "get it right the first time". The field of human computer interaction has not yet been, and may never be, able to capture and standardize all the factors that are necessary to ensure good design from the outset.

This paper will focus on a few usability evaluation processes as they were applied to the HP OpenView System Manager product. The methods used were chosen to maximize the return given the time and resource constraints under which product development occurred.

Describing the user and the user environment

The first task in optimizing the usability of a product is to define the expected user of the product and the environment in which that user will be operating. This must be done as early in the product life cycle as is possible, that is, in the requirements definition/planning phase.

In the case of HP OpenView System Manager, the users are considered to be those individuals with operations responsibility which we will call "operators" and those individuals with overall computer system management responsibility which we will call "managers". The latter are variously referred to as System Administrators, System Managers, or Data Center Managers by different organizations.

The environment is either a high-end data center or a low-end office. We further defined this environment in terms of the number of users, number of disk and tape drives, number of printers, networking, software, applications, etc.

Thus, three distinct profiles for the user of the OVSM product could be identified:

- current HP 3000 customers (low-end through midrange)

- current IBM mainframe data center customers (new high-end business)
- new low-end customers

It was assumed that the operator and manager are not separate at the low end. Consequently, there were five identifiable user groups.

Table 1. OVSM User Groups

Customer profile	User group
Current HP 3000 MPE/iX customers - upgrading or consolidating	■ HP 3000 experienced managers ■ HP 3000 experienced operators
Current IBM mainframe Data Centers - offloading or replacing the mainframe	■ IBM mainframe experienced managers ■ IBM mainframe experienced operators
New low-end customers	■ Novice nonfull-time administrators

As an example, the following table lists the characteristics of two of the above user groups.

Table 2. Characteristics of Representative OVSM User Groups

HP 3000 experienced managers	HP 3000 experienced operators
■ current full-time system manager ■ has had formal HP 3000 system manager training ■ may have CS or similar degree ■ more than five years of MIS experience ■ more than two years of HP 3000 system management experience ■ has performed the following tasks: * managed accounts * managed storage requirements * installed and configured systems and peripherals (disk drives, printers) * updated the MPE/iX operating system * written scripts to automate procedures ■ may or may not have HP OpenView or MS-Windows experience ■ may or may not have PC and/or DOS experience ■ may or may not have LAN experience ■ may or may not have HP AdvanceLink experience	■ Current full-time operator for an HP 3000 ■ has at least six months of operations experience ■ familiar with the current HP 3000 console commands: * CTRL-A commands * LIMIT, ABORTJOB ■ has performed the following tasks: * performed system backups * responded to user requests * managed tapes for backup and for file restore * streamed jobs * managed printing ■ may or may not have HP OpenView or MS-Windows experience ■ may or may not have PC and/or DOS experience ■ may or may not have LAN experience ■ may or may not have HP AdvanceLink experience

Describing the tasks the user will want or need to perform

The next major task required for optimizing the usability of a product is to describe the tasks the user will want or need to perform using the product.

Some of the most important tasks we expected the OVSM user to need to accomplish are:

Overall learning and use

- learn to understand and use the OVSM product
- obtain assistance with the OVSM product when needed
- perform standard system console activities such as tape replies, system startup and shutdown

Accomplishing basic message handling tasks

- start and stop the OVSM software on the PC and on the HP 3000
- identify subsystems requiring attention
- respond to an alert
- read an event message and respond appropriately
- clear a system or subsystem icon color
- delete an event message from OVSM
- annotate an event message
- view event messages from several subsystems
- save an event message for later review
- review a saved message
- create reports of event message activity

Administering OVSM

- set up the OVSM system
- customize the OVSM PC screen
- configure the OVSM event message IMAGE database
- add/delete/change capabilities of OVSM users
- set up a redundant master console
- set up task-based consoles
- backup and archive event messages and the event message database
- troubleshoot problems with the OVSM product

Accomplishing advanced message handling tasks

- redirect event messages to other subsystem icons
- add codes to HP 3000 messages to allow OVSM trapping
- allow OVSM to monitor application event messages
- set up automatic response to event messages (scripted response)

Setting product usability objectives

The next key step in ensuring optimum usability of a product is to set usability objectives or goals for the product. These objectives must focus on ensuring that the product will function to facilitate the expected user's accomplishing the tasks they are projected to want or need to accomplish. The objectives must be measurable.

The overall usability objectives for the OVSM product are listed below. Note that these objectives do not include quantifiable measures. For example, the second objective, OVSM is easy to learn, would need to state what the user can learn to do, with what accuracy, and in what period of time in order to include quantitative measures. At the time we set the objectives, we realized that we did not have sufficient data to be this specific and that our overall purpose was to identify and eliminate the most obvious problems the users might encounter while using the product. As you will see, working with these more general objectives did not preclude our finding and repairing a large number of usability defects.

The objectives were:

- OVSM is accessible, that is, it is obvious that OVSM is a part of the system and is easy to set up, configure, and customize for the user's environment.
- The user's first impression of the OVSM product is that it will be better than previous console interaction modes. The user perceives that at least some tasks will be easier to perform using OVSM as compared to previous console interaction modes.
- OVSM is easy to learn.
- OVSM is compatible with existing console interactions and knowledge can easily be transferred from previous ways of performing tasks to the OVSM way.
- OVSM icons and the icon map are perceived to be intuitive. The screen presentation and the way it is used map to the user's mental model of tasks they must perform.
- The user can determine what process is affected when icon activity occurs.
- The user can easily create reports of message activity.
- Routine administration of OVSM can be easily accomplished.
- OVSM troubleshooting is efficiently performed.

- Using OVSM is perceived as an improvement in productivity. Using it requires less effort than performing the same tasks the "previous way".
- 100% of the required system administration tasks can be performed using OVSM.

Deciding to do a usability review

We do a usability review when we believe that there may be a risk of usability defects or issues with a product and that these defects could significantly impact the expected user of the product. A usability defect is anything that interferes with the user's ability to efficiently and effectively use the system to accomplish his or her tasks.

The purpose of the usability evaluation is to identify defects in either the product or the supporting documentation that would cause a key risk to user satisfaction in either the initial release or subsequent releases. A number of considerations led us to conclude that we should evaluate the OVSM product for usability. Among these were the following.

- The product is PC-based and was to be the first PC-based console for the HP 3000. We felt that is was very important that the user perceive that the incremental cost due to the additional hardware be justified by the benefits the user received from using the product.
- Users' expectations of graphical interfaces (GUIs) are somewhat higher than their expectations of command line interfaces with respect to being intuitive and self-explanatory. We felt that we must ensure that these expectations were satisfactorily met.
- In the case of the OVSM and OVC, usability did not become a part of the product development team until late in the development cycle precluding going through some of the usability evaluations that normally occur in the early phases (planning, investigation, design, etc.) of product development.

Deciding on a Usability Evaluation Plan

The next step is to decide what types of usability evaluations to perform on the product. In the case of OVSM, as with most industrial products in today's rapidly changing marketplace, we were constrained by both schedule and resources. Combining these constraints with the user and environment descriptions, user task descriptions, and usability objectives, we determined that we should proceed with several phases of evaluation. These were, in order:

- heuristic evaluation by a Human Factors (HF) or Human-Computer Interaction (HCI) engineer
- user performance walkthrough of the product
- locator walkthrough of the documentation
- product usability inspection by a multi-functional team of engineers

The first three of these are complete and will be described in the remainder of this paper.

Usability evaluation schedule

The approximate schedule for the project was as follows:

Table 3. HP OpenView System Manager Usability Evaluation Schedule

Milestone	Approximate Month
Begin usability involvement in project team	1 (by definition)
Usability plan complete (includes user profiles, user environment, task descriptions, usability objectives, preliminary schedule and resources)	3
Heuristic evaluation by HFE	4
User performance walkthrough test plan	5
User performance walkthrough	6
User performance walkthrough preliminary report	6
User performance walkthrough final report	7
First product release (OVC only) final software submittal	7
Second product release (OVSM and OVC) final software submittal	11
Locator walkthrough test plan	12
Locator walkthrough	13
Locator walkthrough preliminary report	13
Locator walkthrough final report	15
Third product release (OVSM and OVC) final software submittal	~22

Heuristic evaluation of HP OpenView Console

A **heuristic evaluation** is an inspection of the product's user interface with respect to established usability principles and the usability goals for the product. The inspection can be carried out by one or two HF (Human Factors) or HCI

(Human-Computer Interaction) engineers or by a team of product developers, preferably with some guidance from an HF or HCI engineer. The inspector(s) evaluate the product for compliance with the usability principles and goals as they apply to the expected users in their environment performing the most important tasks they are expected to perform.

We chose this method for the first phase of evaluation for several reasons. One of these was that this could be done rather quickly so as to maximize the possibility of fixes before the product's first release. Another was that we wanted to find and fix the most significant defects before expending the time and money required for a full user performance walkthrough.

For this first evaluation, since the first release of the product was projected to be used primarily by users consolidating HP 3000s or moving from a mainframe, we did not consider the needs of the low end administrator.

We identified 18 usability defects by heuristic evaluation. Nine of these were related to the appearance of the default screen. All of these were fixed prior to the user performance walkthrough. Some of the defects that were fixed as a result of this evaluation were the following (refer to Figures 1 and 2):

- screen real estate was not used as efficiently as possible
- some icons were not representative of the subsystem grouping they were intended to represent
- some icons were not expected to be universally used
- the icon category names were unnecessarily long
- the center icon served several purposes. Any event message without an identifier was redirected to a new "no-ID" icon
- OVSM system components were not broken out from HP 3000 subsystems

Seven of the remaining nine usability defects were partially addressed prior to the user performance walkthrough and all were addressed as part of the response to the user performance walkthrough results (refer to the next section).

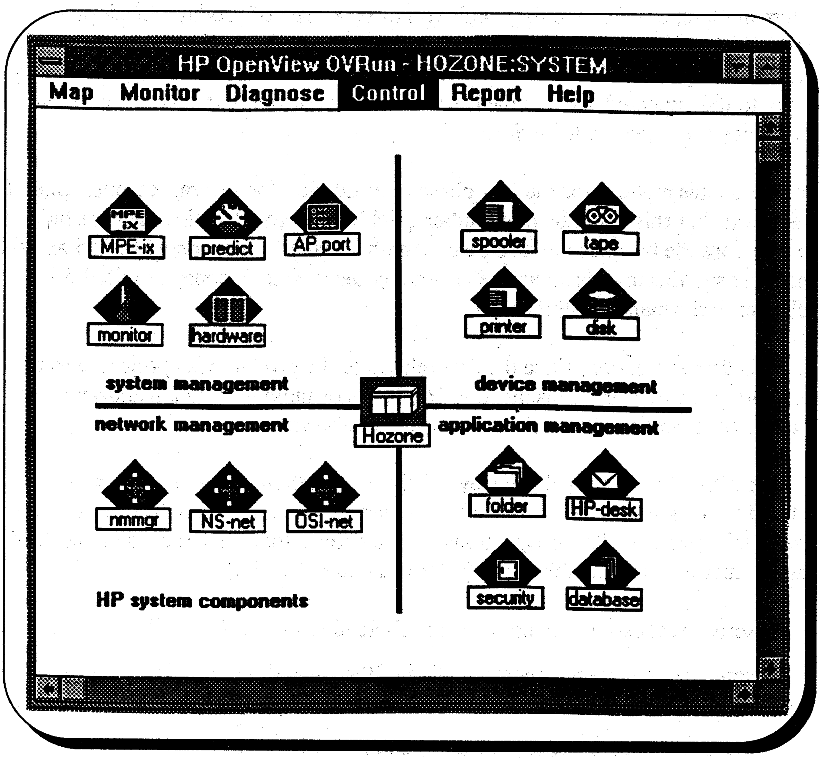

HP OpenView OVRun - HOZONE:SYSTEM
Map
Monitor
Diagnose
Control
Report
Help
MPE-ix
predict
AP port
monitor
hardware
spooler
tape
printer
disk
system management
device management
network management
Hozone
application management
nmmgr
NS-net
OSI-net
folder
HP-desk
HP system components
security
database

Figure 1. Default OVC screen before heuristic evaluation.

User performance walkthrough of HP OpenView Console

A **user performance walkthrough** is a controlled lab test in which representatives of the expected user population are observed performing representative tasks using the product or a prototype of the product. The observer and a video camera record user actions and comments in order to identify usability problems. Users are encouraged to "think aloud" so that their opinions of the product, approach to the tasks, line of thinking, and the logic behind their errors are as obvious as possible. Users are prompted for their opinions during the test and are debriefed following completion of the test.

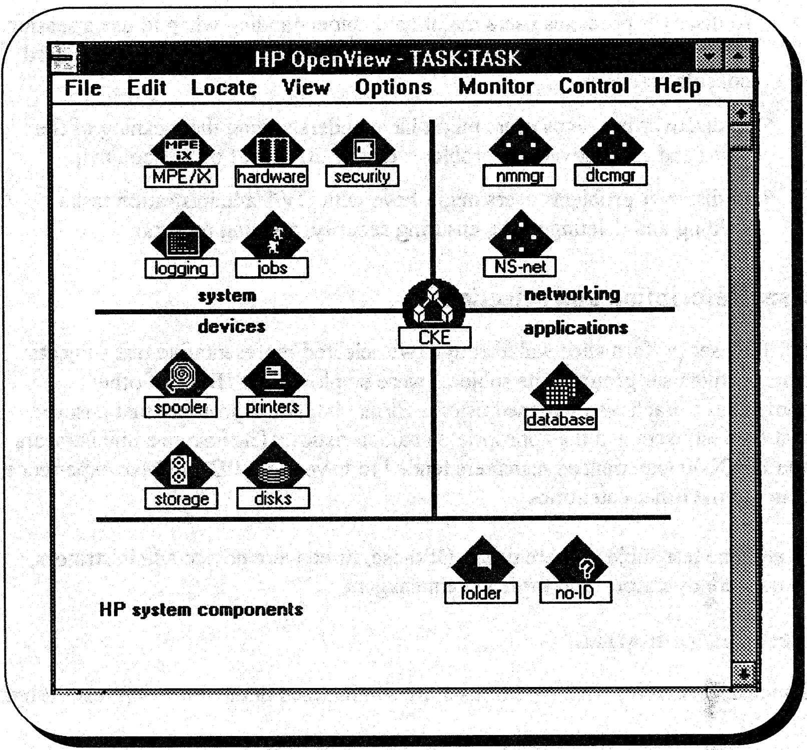

HP OpenView - TASK:TASK
File
Edit
Locate
View
Options
Monitor
Control
Help
MPE iX
MPE/iX
hardware
security
nmmgr
dtcmgr
logging
jobs
NS-net
system
networking
CKE
devices
applications
spooler
printers
database
storage
disks
folder
no-ID
HP system components

Figure 2. Default OVC screen after heuristic evaluation.

The user performance walkthrough is used to diagnose problems that the user has with the product and the documentation including learning problems. Often by analysis of the paths the users take to accomplish tasks, recommendations to fix usability defects become obvious.

Test objectives.

The test objectives for the user performance walkthrough of OVC were:

- to discover reasons users might have for a negative initial impression of the product
- to discover problems users might have in understanding how to achieve basic message handling tasks

- to discover problems users might have understanding when to use a session window and when to use the console window when carrying out standard console activities
- to discover problems users might have understanding the meaning of the icons and to discover any problems due to the layout of the icon map
- to discover problems users might have with OVC administration tasks (adding and deleting users, ensuring security, creating reports).

User description and selection.

For the user performance walkthrough, we selected representative test subjects from all five user groups. The subjects were employees of HP or of other companies. Each was screened using a formal interview protocol that ensured that the test users had the appropriate characteristics. The novice administrators and HP 3000 experienced managers tended to have more PC and Mac experience than did the other categories.

In all, nine test subjects were used. Of these, three were novice administrators, three were operators, and three were managers.

Test design matrix.

The tasks tested for each type of user are summarized in the following test design matrix.

Table 4. OVC User Performance Walkthrough Test Matrix

Task	User group
Console and message management tasks ▪ start and stop OVC ▪ read messages ▪ issue console only commands ▪ issue system management commands ▪ clear colors ▪ create, append and view annotations ▪ save messages in the folder ▪ manage message views	▪ HP 3000 experienced operators ▪ IBM mainframe experienced operators

Task	User group
Low end administrator/operator tasks ▪ start and stop OVC ▪ read messages ▪ issue console only commands ▪ issue system management commands ▪ clear colors ▪ create, append and view annotations ▪ save messages in the folder ▪ manage message views ▪ add and delete OVC users	▪ novice administrators
Message management and setup tasks ▪ start and stop OVC ▪ read messages ▪ issue console only commands ▪ issue system management commands ▪ clear colors ▪ create, append, and view annotations ▪ save messages in the folder ▪ manage message views ▪ create message reports ▪ configure the EML database ▪ add and delete OVC users	▪ HP 3000 experienced managers ▪ IBM mainframe experienced managers

Many of the above activities were simulated using command files. In addition, there were two programs created, one that sent an informational event message to be treated as a normal HP 3000 event by OVC and the other that sent an event message requiring a yes or no reply using the standard REPLY command. There were 27 task scenarios utilized in the tests, 14 of which were designed for all test user categories, and 13 of which were designed for managers only.

Test procedure

There were two distinct test sessions which were one half day for operators and novice administrators and one full day for managers. The tests were conducted in the HP Cupertino usability lab and were videotaped.

The test session began with a user orientation in which the subjects were told what would happen during the test and what was expected of them. They were introduced to the testing team and to the lab and its equipment. Then they were asked to fill out the necessary forms for confidentiality, videotaping, and so on.

Just prior to beginning the task scenarios, each user was given an orientation to the product itself (and to the PC, if required). They were also given an introduction to the task scenarios that included a detailed description of the environment in which they would be operating.

Following the orientation, the test users were asked to complete the task scenarios, one by one. During each task, the users were reminded to "think aloud" and their comments and actions were recorded for later analysis. The test users were asked to perform representative tasks, one at a time, just as if they were tasks they needed to perform in their real jobs. They were also asked for feedback during the test.

After they had attempted all the tasks, they were given a post-task attitude questionnaire. They were also debriefed to determine their current level of understanding of the product and their suggestions for improvement.

A typical task scenario, as seen by the user, is the following:

> **Background**
>
> Your system performs unattended backups using a few regularly scheduled jobs:
>
> FullBack handles full backup once a week and PartBack performs a partial backup every night. The backup device is the Optical Library which stores files on optical disks rather than tapes.
>
> These jobs handle the repeatable steps of setting up a backup and so on. These jobs have been written so that they send messages to the console when they have completed with details of the number of volumes used to complete the backup.
>
> **Instructions**
>
> - Check to see if the messages from last night's partial are still available for review.
> - If everything is fine, delete the messages relating to the backup.

Figure 3. Typical task scenario used in OVC user performance walkthrough.

Defect descriptions

The definitions of severity we used to classify the usability defects discovered were the following:

Critical	A critical defect is any one that renders the system unusable by the typical customer, therefore severely affecting the customer's daily operations. "Unusable" means that customers can't or won't use an essential part of the system. The customer may experience real or perceived data loss or corruption.
Serious	A serious defect is any defect that makes the essential parts of the system difficult, but not impossible, to use. The customer's daily operation is still possible, but is affected by severe restrictions, significant delays, or much frustration and dissatisfaction.
Medium	A medium defect is any defect that causes difficulty to a typical user but does not have a serious impact on the functionality of the essential parts of the system. It may cause some user frustration and dissatisfaction with either essential or nonessential parts of the system.
Low	A low defect is any defect that only affects the user's daily operation by causing minor inconvenience, frustration, and/or dissatisfaction. The customer can circumvent the problem.

There were a total of 94 usability defects found during the user performance walkthrough. These were found in the OVC PC software, HP 3000 software, the user interface, the help facility, and the learning products.

Additionally, 22 software bugs were found opportunistically during the OVC usability testing.

Defect repair

A report was prepared in which each defect was described, along with recommended fixes. An example will be used to illustrate this process. One of the usability defects that was classified as serious was:

Only one Events Browser (a list of HP 3000 event messages from a single subsystem grouping) can be open at one time and users have to explicitly close the open Events Browser before opening another. The problem statement explained that when a user is using an Events Browser and they wish to examine a message that has just arrived in another subsystem grouping or to annotate a message that belongs in another subsystem grouping, they must first perform two steps, namely close the open browser and open the second browser. Additionally, the error message that appeared on the screen when the user attempted to open a browser while one was already open was not clear to the users.

The recommended solution was to allow multiple browsers to be open at one time or, as a step in that direction, to automatically close the open browser when the user requests another browser to be opened.

For the next release of the product, the error message has been clarified and when a second browser is requested to be opened, the open browser is automatically closed.

Locator walkthrough test of HP OpenView System Manager on-line help

A **locator walkthrough test** is similar to the user performance walkthrough described in the previous section. The major difference is that the locator walkthrough is used to diagnose problems the user has with locating information as opposed to problems with performing a task. Other than this difference in overall objective, the test is performed in the same manner as the user performance walkthrough.

The test we performed was a qualitative test of three prototype help structures with different tables of contents or entry points.

Test objectives

The objectives of the test were to, prior to developing the majority of the text and hyperlinks, discover potential problems in:

- the overall help structure
- the specific content of the navigation only pages and other location aids
- the format of the task-based information pages
- the error message approach using search and error numbers

Additionally, we wished to provide design constraints to be used in the design of the final help facility structure and to identify general guidelines for on-line help structure development. We will not address the latter goal in this paper.

The test did not cover any specific topic content such as conceptual material or task instructions since the majority of this material had not been developed at the time of the test.

User description and selection

We selected representatives of the operator and manager groups for testing. All of the subjects were HP employees who are full time operators or managers of an HP 3000 system. The subjects were selected based on their qualification by use of a background experience interview questionnaire. Since the major daily user of the OVSM product is expected to be the System Operator, more subjects that fell into this category were tested (6 operators and 3 managers). Most of the nine subjects (seven) had no prior experience with OVSM.

Test design matrix

There were 43 tasks tested during this test. Of these, 34 applied to all user types and the remaining (administrative tasks) applied only to the managers tested.

The tasks tested covered the following areas (an example task for each area appears in parentheses):

- event monitoring concepts (what the maps represent)
- event monitoring tasks (viewing a list of all events)
- event message handling concepts (what kinds of terminal/console connections are possible)
- event message handling tasks (deleting a message from a browser)
- reporting tasks (creating a printed report of messages using the system printer)
- customizing tasks (creating an automatic response for an event)
- administration concepts (OVSM login versus HP 3000 login)
- administration tasks (adding a OVSM user)
- troubleshooting tasks (locate information on OVSM error messages)

Test procedure

The tests took two to five hours per person. They were conducted in a standard workspace cubicle and were videotaped.

The test session began with a user orientation, introduction, forms, and product and environment orientation just as was described above for the user performance walkthrough. Note that in this case the product was OVSM as opposed to OVC and only the help system was evaluated.

For each task scenario, the users were asked "Where in the help system would you look to find out" followed by the subject of the task. For example, "Where in the help system would you look to find out what happens to messages that you delete from the message list display?"

The users were given about two minutes to complete each task. It was explained to them during the initial orientation that if the help system was as effective as we thought it should be, they should be able to locate the information they were seeking in about one minute or less. Any longer times indicated a defect in the help system that we needed to repair.

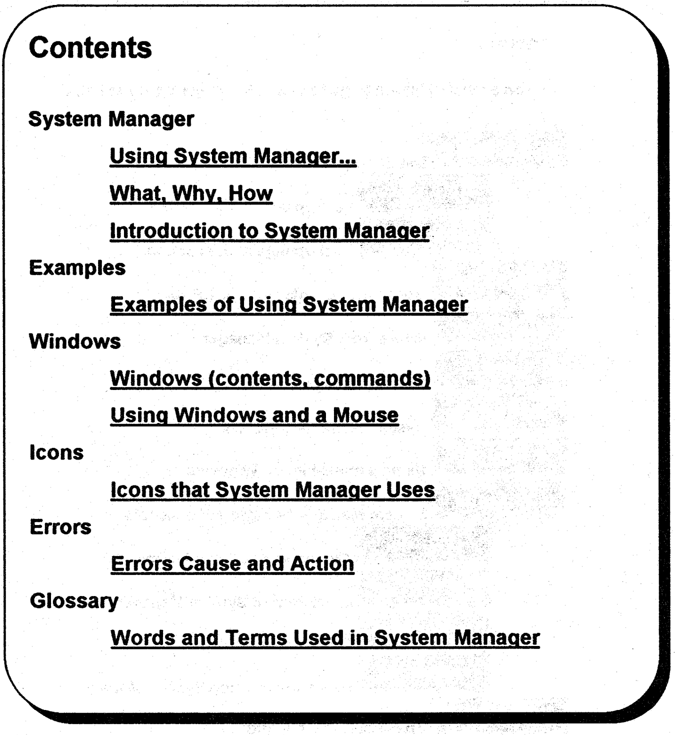
Defect description and repair.

Locatability defects were identified for all but three of the 43 task scenarios tested. Additionally, 12 overall usability defects that incorporated the specific scenario defects were identified (two critical, four serious, five medium, and one low). Again a report was prepared in which each defect was described, along with recommended fixes. An example illustrates some of the defects found and one iteration of implementation of fixes.

The top level table of contents for one of the prototypes of the help system tested was as shown in Figure 4. After testing, the top level prototype was revised to respond to recommendations for fixing the usability defects discovered. One iteration of the revised prototype is shown in Figure 5.

Some of the usability defects found in the tested prototype that were repaired in the revised prototype were the following (refer to figures 4 and 5):

- The test users were all confused over the concepts of event notification, events, errors, and messages. They did not find the "messages" entry under "Using System Manager...".
  Fix. A top level category for Messages (events on your HP 3000) was added and the top level Errors category was clarified to explain that it covered only errors from OVSM. An extensively indexed "Quick Tour" of OVSM was added to give the users an overview of the product.
- The test users had a great deal of difficulty deciding which entry to choose and what was explodable at the first (Contents) entry point to the help facility.
  Fix. The explodable items were turned into buttons with adjacent explanatory text. The categories available at the top level were revised extensively.
- The test users did not identify the task-oriented category labeled "Using System Manager..." as a useful entry point.
  Fix. The task-oriented category was relabeled "Tasks".



Contents

System Manager

Using System Manager...

What, Why, How

Introduction to System Manager

Examples

Examples of Using System Manager

Windows

Windows (contents, commands)

Using Windows and a Mouse

Icons

Icons that System Manager Uses

Errors

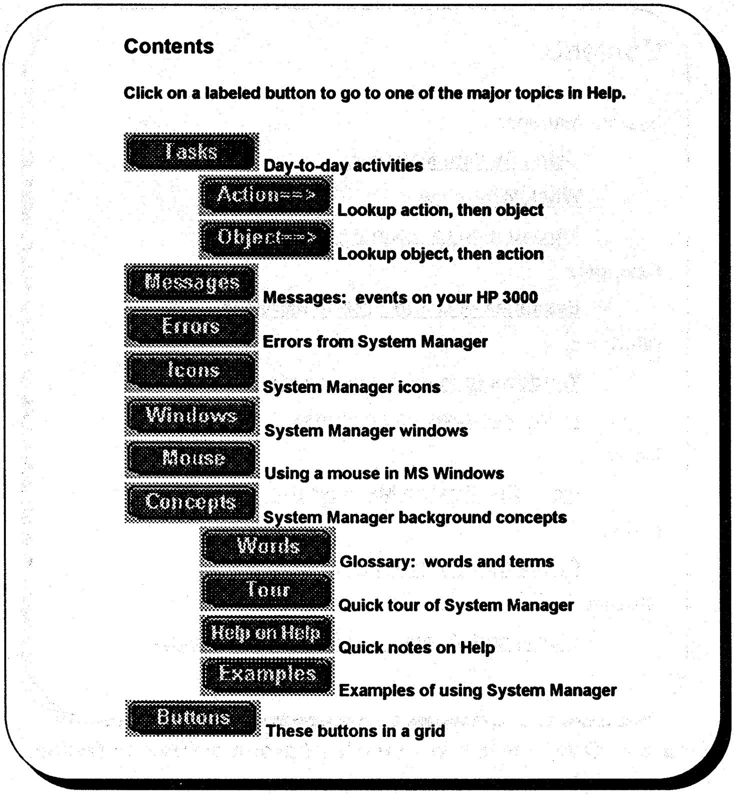
Errors Cause and Action

Glossary

Words and Terms Used in System Manager

Figure 4. OVSM help system top level prototype used in testing.

- The test users were confused about the meaning of the category "Windows". They could not determine if it covered Microsoft Windows or OVSM windows.
  Fix. The "Windows" category was clarified to make it clear that it covered OVSM windows and a top level "Mouse" category was added to cover the more general Microsoft Windows and the use of a mouse.
- The test users did not find the "Examples" category useful.
  Fix. The "Examples" category was moved down the list.



Contents

Click on a labeled button to go to one of the major topics in Help.

Day-to-day activities
Lookup action, then object
Lookup object, then action
Messages: events on your HP 3000
Errors from System Manager
System Manager icons
System Manager windows
Using a mouse in MS Windows
System Manager background concepts
Glossary: words and terms
Quick tour of System Manager
Quick notes on Help
Examples of using System Manager
These buttons in a grid

Figure 5. OVSM help system top level prototype after revision to repair usability defects.

Summary and Conclusions

The paper describes three processes used to evaluate the usability of the HP OpenView System Manager product and documentation. These were heuristic evaluation, user performance walkthrough, and locator walkthrough. The methods were used in different phases of the product development cycles and each resulted in identification and repair of a large number of usability defects prior to release of the product. By involving appropriate test users in product design and development, it is possible to greatly increase a product's ability to satisfy user needs in the user's environment.

Paper#5014

Migrating PowerHouse® to UNIX

Ben Foulkes
Cognos Incorporated
3755 Riverside Drive
Ottawa, Ontario, Canada

This document is written for the customer who has a UNIX system, or is considering the purchase of one either to supplement an existing configuration or eventually to replace it. This document describes how easy the transition to UNIX can be with PowerHouse. There are many different platforms on which PowerHouse can reside; however this document primarily oulines the migration steps from HP MPiX. Since MPE/V is similar in most cases, most of the context of this document should apply as well.

Advantages of migrating to UNIX

The UNIX operating system has become a very popular place for business applications, mainly because it offers a great deal of power at a relatively low cost. The fact that it is so popular has opened up the marketplace to competition, ensuring top quality products. This means that PowerHouse will be pushed to the limit during our testing phases to ensure a top quality product.

Networking:

Since the UNIX environment was "born" to network, having many UNIX machines working together is no longer a network nightmare. It is essentially the same as simply hooking together two PCs with ethernet cards.

Databases:

A multitude of databases are supported on UNIX. PowerHouse will support ALLBASE, InterBase and soon it will also support Sybase. Of course, there is still support for CISAM and unixio (stream) files.

Support:

Cognos' entire support staff are all trained in the UNIX operating system. There is always someone available to help you through the migration.

Disadvantages

Although there are few disadvantages, it should be noted that UNIX does not support Relative, Circular, or Message files. Since many techniques have been developed around using message files, workarounds are available using features specific to the UNIX operating system. Another issue to consider is that an Image database that was created on MPE must be accessed through Turbo Connect or IMAGE/SQL (an IMAGE SQL gateway for ALLBASE).

Planning

One of the most important aspects when planning a conversion to UNIX is to make adequate provision for education and training. Without this, it is extremely difficult to make reliable estimates on the amount of work and length of time it will take to complete a conversion. Further, proper vendor education often adds value to the course material by giving tips and suggestions based on real-life situations from the instructor's experience.

If you intend to convert a large application, choose a suitable "module" or subset of the application for a "proof of concept" conversion exercise. Going through the complete conversion cycle from start to end will allow you to identify relevant issues and gain experience of what is involved in the process. You will be in a position to plan more accurately and to make confident estimates on the process for the entire application.

Provision should be made at each stage in the conversion process for adequate testing, both in isolation and in combination with previous stages. Data used should be a copy of the real business data. User load testing and timings should be performed as a reflection of real production use, not just a simulation.

Cognos has a range of education and consulting services which are appropriate to planning and converting PowerHouse applications to UNIX.

Converting the PDL dictionary

Someone who has looked into the conversion may think "How on earth am I going to convert my Image database?". To be certain, some chopping and hacking will have to be done on the PDL dictionary.

If you are still using QDD dictionary, it will have to be converted. On MPE/iX, you can do this in about 10 lines with the help of Qshow. On MPE/V - version 5.36 will have the same functionality.

```
QSHOW
>SET LANGUAGE PDL
>SET SECURITY
>SET PASSWORD
>GENERATE ALL                 {This generates a file called qshogen}
>EXIT

PDL
>CREATE DICTIONARY PHD
>USE QSHOGEN
>LOAD
```

The task of converting to PDL from QDD is complete pending the result of zero errors returned from PDL. If there were errors, you will have to debug the text file "qshogen" for the errors. If you need help, call Customer Support or refer to the PDL manual. Once you have a PDL dictionary, you can begin your conversion.

Converting an application is straightforward when the underlying file systems on the host and target machines are very similar, because structural and program specification changes can be kept to a minimum. If you do not currently use a relational database system with your application(s) but plan to do so, skip to the next section, *Converting to a Relational Database.*

PowerHouse supports indexed files using Informix's C-ISAM across all UNIX platforms. C-ISAM can be used as a fully functional target file system for application conversion or as a supplement to a third party database. It should be noted that C-ISAM is not supported across NFS (Network File System) mounted disks. This may be an added incentive to move to a relational database management system, if you are planning a distributed application.

When converting from KSAM to C-ISAM, there are only a couple of differences, one of which is the fact that C-SAM does not support the KSAMXL's REUSE option in PDL. This option would have to be taken out. C-ISAM automatically restructures itself already without this option. Prior to C-ISAM version 5.0, the length of the filename is restricted to 10 characters. This should not cause a problem since, MPE has an 8 character restriction. The following is a "worst case" example of converting from KSAMXL to C-ISAM:

```
FILE KSAMFILE ORGANIZATION INDEXED TYPE KSAMXL REUSE ASCII BLOCKING FACTOR 20 CAPACITY
20000 OPEN "KSAMFILE"
RECORD KSAMFILE
ITEM ITEM1 DATATYPE NONIEEE FLOAT
ITEM ITEM2
INDEX ITEM1
SEGMENT ITEM1
```

Since there is no specifiable blocking factor for C-ISAM and files expand dynamically, BLOCKING FACTOR and CAPACITY are not necessary. Also, data types such as NONIEEE are specific to MPE. The PDL for KSAM can be converted to C-ISAM as:

```
FILE CISAMFILE ORGANIZATION INDEXED TYPE CISAM OPEN "CISAMFILE"
RECORD CISAMFILE
ITEM ITEM1 DATATYPE FLOAT
ITEM ITEM2
INDEX ITEM1
SEGMENT ITEM1
```

Note that if you have indexes that reside within another index, KSAM is able to use the outside index as the access path for the sub-indexes. With C-ISAM, a separate index is create for each. This factor will not affect anything from a coding perspective but should be acknowledged.

If you are converting the Image database to an Indexed file system, there are some tools that can make the job very simple. With PowerHouse PC, there is a tool called "XPDL". This tool will convert all references to an image database to be indexed. All security and application security classes must be taken out of the dictionary prior to this since Powerhouse PC does not support security.

If you are currently using Image and/or KSAM, but wish to convert to RDBMS, take note of considerations discussed in the next section.

Converting to a Relational Database Management System

The current release of PowerHouse 703 supports the use of INTERBASE and ALLBASE relational databases. With the next release of 723, Sybase has been added to this list. The features offered by a relational database and a simple indexed structure are significantly different. Therefore, the same approach cannot be used for both, whether developing in PowerHouse or a traditional 3GL.

You may want to consider using a more powerful tool for converting to an RDBMS. With PowerDesigner, you can re-engineer your existing PDL into the PowerDesigner database. Once the all relationships have been verified within the database, the database type can be changed from image to relational. After this, the new PDL and SQL can be generated. Small applications can be converted in a matter of hours using this method.

A PowerHouse application originally generated from PowerDesigner should make conversion to a relational database significantly simpler and quicker than a manual conversion from PDL to SQL.

If your application was initially generated from PowerDesigner, the application analysis and design tool from Cognos, the Database Design Diagrams will allow the choice of generating ANSI-standard SQL DDL (data definition language), or Borland InterBase DDL including trigger syntax for referential integrity.

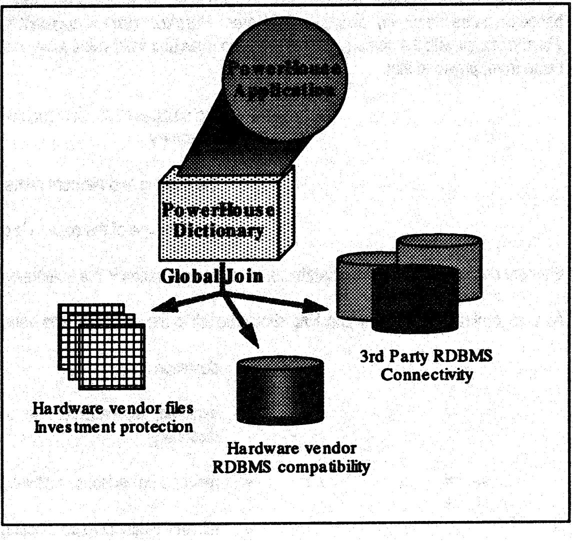
If your application hasn't benefited from any type of structured design, then data redesign may be necessary. Traditional file systems do not require the underlying data structures to be normalized, which means that they do not need to meet the rules for First Normal Form (1NF), a mandatory requirement for relational databases. Irrespective of file system, it is recommended that data be normalized to at least Third Normal Form (3NF).

A full description of the decomposition and normalization process is not included here. The PowerDesign education course, which integrates industry-standard design techniques with techniques optimized for PowerHouse 4GL, covers this subject in depth.

PowerHouse Dictionary

The PowerHouse dictionary is at the core of an application. It gives the ability to access and globally join data from a variety of databases. For example, data from an ISAM file and from a table in a relational database can be updated on a single screen, or combined in one report.

PowerHouse applications access their data through the PowerHouse Dictionary giving them a single, unified view of diverse file systems and databases.


PowerHouse Application
PowerHouse Dictionary
Global Join
Hardware vendor files
Investment protection
Hardware vendor
RDBMS compatibility
3rd Party RDBMS
Connectivity

In your source system, entire file definitions and element information were held in the dictionary, but this information must now be distributed between the database and the PowerHouse dictionary.

Database definitions are not repeated in the PowerHouse dictionary, as tables, column definitions and security are all obtained directly from the system tables. These internal definitions are often referred to as "metadata".

PowerHouse treats these system tables as a subdictionary. If new tables are added or existing tables are changed, PowerHouse picks up the new information during compilation. There are no redundant table definitions to keep synchronized.

The PowerHouse components obtain extensive information from element definitions in the PowerHouse dictionary to check for valid values and to control the presentation of data on reports or screens. For example, column headings, screen labels and display formatting can all be defined in the dictionary. Reasonable defaults can be derived from the column definitions in the system tables of the database. These defaults may not be appropriate for your application; therefore, the PowerHouse dictionary can be used to enhance relational column definitions with element definitions based on name match. For example, an element *PartNumber* will augment any relational column named *PartNumber* with the same set of presentation rules and valid value rules, regardless of which table or database it came from, provided that:

- the database and element have been declared in the same dictionary
- the column and element name match
- the datatype of the column is compatible with that of the element

Element definitions bind a multi-database, multi-file application into a seamless unit.

As a general rule, all integrity checking should be left to the database. The benefits of this are:

- definition consistency
- minimum duplication of rules in InterBase and the PowerHouse dictionary
- reduced PowerHouse edit validation processing
- minimal dictionary specification required
- cleaner and easier porting to other platforms using the same database

It is recommended that the validation performed by "lookup" processing and field validation rules should remain in the PowerHouse application to provide the end user with interactive dialogue. This provides immediate feedback to the user at the point of error during data entry. In contrast, column validation rules specified in the RDBMS are enforced when a record is being stored or modified. An input error may only be detected when trying to commit a transaction, resulting in the rollback of the transaction and a potential loss of screen data.

One of the biggest changes that your PDL dictionary will undergo is size. When accessing a relational database, all you really need is 3 lines of code in your dictionary.

```
CREATE DICTIONARY PHD
FILE MYDATABASE ORGANIZATION RELATION TYPE [STAR|ALLBASE|...] OPEN ...
LOAD
```

The PowerHouse products will automatically pick up the tables or relations that are within the database.

An important part of converting to a relational database is to revise the indexes that will be defined. As with ISAM file systems, indexes are an overhead, particularly non-unique ones. Indexes serve two purposes: enforcing data integrity by making the record unique, and providing the database query optimizer with a faster method of record retrieval. However, the database may not use an index if it is not an optimal retrieval path for the data, so if the relation is small, you may not require the index unless it is explicitly required by the application. It should be noted that relational databases like InterBase can make use of more than one index at a time for optimizing retrieval, whereas non-relational systems can't. However, too many indexes that reference a single column can degrade performance.

Integration and optimization

If you decide to take full advantage of the capabilities of a relational database rather than using it as a simple indexed file system, the following section describes how it is possible to provide additional application optimization and database integration.

Further opportunities for optimization will be available in PowerHouse 4GL with embedded SQL statements and an enhanced relational transaction model in the next release of PowerHouse on UNIX.

Triggers

Today, most advanced relational database systems support triggers. Triggers are applied during an application's operation to enforce business rules or logic. Being a piece of code that is stored once within the database, triggers simplify maintenance and ensure that all applications, whether written in PowerHouse 4GL or a 3GL, obey the same rules.

Triggers can be used to replace some of the procedural processing previously implemented within the PowerHouse application. The following examples are based on Borland InterBase:

This trigger addresses referential integrity by ensuring that there are no outstanding orders on a part before it is deleted. The "abort 1" syntax forces the transaction to abort/rollback and report an error on this trigger to the user.

```
define trigger erase_parts for parts_master
pre erase 0:
begin
        if any o in orders with o.part_number = old.part_number
        then abort 1;
end;
end_trigger;
message 1:"There is an order which requires this part.";
```

This trigger assigns incremental part numbers, records the date entered and who entered it.

```
define trigger new_part for parts_master
pre store 0:
begin
        new.part_number = gen_id( part_numbers, 1 );
        new.date_entered = "now";
        new.date_entered_by =  rdb$user_name;
end;
end_trigger;
```

This trigger performs a cascading delete which prevents orphaned records. When an order is deleted all related order lines are also deleted.

```
define trigger delete_orders for orders
pre erase 1:
        for o in order_lines with o.order_number = old.order_number
        erase o;
end_for;
end_trigger;
```

Views

A view is a "virtual" table that consists only of a stored definition rather than stored data. A view derives data from other table or views each time it's requested. To a user, it is indistinguishable from a regular stored table. Views can be used to impose access restrictions at a field level, or to simplify complex data access and selection processing. In addition, view definitions are held within the database, providing performance gains, consistency and simplifying maintenance.

This view will retrieve all retired employees.

```
define view retired_employees of e in employees
 with e.retired_date not missing
        e.employee_id,
        e.employee_name,
        e.retired_date,
        ...
```

This view will retrieve all order that have received payments.

```
define view order_payments of o in orders
 cross p in payments with o.order_number = p.order_number
        o.order_number,
        o.order_date,
        p.payment_date,
        p.payment_amount,
        ...
```

Computed Fields

Computed fields allow you to define business functions and calculations within the database. They are stored once and reused. Computed fields define information or algorithms such as "order line amount." A computed field allows you to define what comprises "order line amount", e.g. *order quantity * product price*, but doesn't store the value in the database; it simply returns the calculated value at the time of the request. This means space is not wasted and maintenance is simplified.

These computed fields calculate the length of service for an employee and the number of illness days taken.

```
define relation employees
        employee_id,
        date_joined,
        ...
        service computed by ( "now" - date_joined ),
        days_ill computed by ( count of i in employee_illness
          with i.employee_id = employee_id );
```

These computed fields calculate how a PowerHouse optional linkage or SQL outer join can be emulated for "foreign key" expansions.

```
define relation employees
        employee_id,
        ...
        branch_code,
        branch computed by ( first b.branch_name from b in branch
        with b.branch_code = branch_code else 'Invalid branch." );
```

Triggers, views and computed fields offer a way to reduce application complexity and simplify maintenance. Proper load testing and timing should be done when making such changes to ensure performance meets expectation.

Advanced Relational Interface/SQL support

With the up coming release of 723, PowerHouse will allow for embedded SQL. This means that the developer will have complete control over transactions. One reason for SQL support is performance. With the ability to use SQL directly within your programs, you will save the CPU time it would normally take PowerHouse to convert to SQL.

A Quiz example:

```
ACCESS INVOICES LINK INVOICE_NO TO INVOICE_NO OF DETAILS
SELECT IF PRODUCT_NO OF DETAILS = MOST
REP ...
GO
```

Might be converted as follows:

```
SQL DECLARE CURSOR MYDATA FOR
  SELECT * FROM INVOICES, DETAILS
  WHERE INVOICES.INVOICE_NO = DETAILS.INVOICE_NO
  AND DETAILS.PRODUCT_NO = MOST
REP ...
GO
```

Network Solutions

With networks becoming a big part of everyones life, your application may require databases to reside on several different machines over the network. PowerHouse can take advantage of this and access data on any system using the technology built right into the database itself. Interbase can access another InterBase database from any other system on which InterBase itself also resides through a TCP/IP network. With this in mind, you could conceivably leave your database on your MPE/iX machine and access it from the UNIX machine. Although on one hand, it would increase network traffic, on the other it would reduce CPU usage on the UNIX machine. Some databases, like Sybase, can have one database residing locally as well as another residing remotely while keeping them in sync by "trickling" the data through the network at such a minimal rate that it causes no network hangups when large transactions are taking place.

Operating System Environment

For the PowerHouse end-user, the application appears and functions the same way, whether it runs under UNIX or a non-UNIX operating system. As we have seen, the developer porting a PowerHouse application will find that the program specification is largely generic. However, any specification which interacts directly with the operating system must be revised for the new environment. The issues to be considered can be broadly categorized as :

- operating system command language
- parameter passing
- case-sensitivity
- directory structures and security
- filename specifications
- program specification changes

Operating system command language

If the application depends heavily on batch and command language processing, replicating this under UNIX will constitute the most lengthy and complicated part of the conversion effort. Proprietary operating systems are feature-rich. Many tools and utilities are provided, and building complex applications is easy for the experienced developer who works within the parameters of these products. In comparison, UNIX is an operating system built by developers for developers. With system call-level features like pipes and input/output redirection, UNIX is quite flexible when compared to most proprietary operating systems. Since input/ouput redirection is also available on MPE/iX, this is one feature that is directly compatible.

The command interface on a UNIX system is known as the Shell. The three most popular are the Bourne Shell, the C Shell and the Korn Shell. Of these, the Bourne Shell is the oldest and the smallest, and has the simplest syntax for Shell programming. Bourne Shell scripts can be executed from within the Korn and C Shells, and these two newer Shells have a number of time-saving features (aliasing, command history, command completion) which make them easier to use interactively. In general, this means that UNIX developers will choose to work within the C or Korn Shells, but write Bourne Shell scripts. PowerHouse reflects this; COMMAND and RUN COMMAND statements will execute in the Bourne Shell. Bear in mind that hard-coded path names and print queue identifiers must also be modified for UNIX.

Case-sensitivity

UNIX is case-sensitive, and you may find it easier and more consistent with the UNIX philosophy to use lower case throughout for your program specifications and filenames. A UNIX filename can be composed of any characters - even unprintable ones - except for '/' and null, which have special meanings to the UNIX shells. PowerHouse appends and looks for lower-case extensions to compiled programs, as it does under proprietary operating systems, although these are considered as nothing more than the last four characters of the filename under UNIX.

Directory structures and Security/Filename specifications

UNIX has a hierarchical directory structure. File access permissions are similar to most proprietary operating systems, which means that the environment where your source application resides may be easily replicated under UNIX. With HP MPE, one might use **ALTSEC** to change security on a file, however on UNIX, **chmod** and **chown** are used for this.

Hard-coded path names in your dictionary and program specifications may be changed to reflect the UNIX equivalent, or you can replace these designations with environment variables. The variable values, and therefore the target files referred to in your application, can then be assigned dynamically. For example, a user and a developer could use the same dictionary, but work with live or test files depending on the values assigned to the environment variables for their session. The PDL statement to assign an environment variable as an open name to a database or file is :

FILE HumanResources ORGANIZATION RELATIONAL OPEN **$varname**

The value assigned to this variable **$varname** might be **usr/hr/data/hr.gdb** for an end user, while for the developer it could be **usr/hr/data/test.gdb**. One warning - the value assigned to an environment variable in a child or sub-process will be lost when the child process and its associated environment terminates.

Program specification changes

A major PowerHouse strength is its tight integration with the hardware vendors' platforms. The product is optimized for each platform to provide the best performance with proprietary file systems, and syntax is provided for the PowerHouse developer in order to take full advantage of the many platform-specific features or utilities available.

The best approach for handling these differences is determined by the intended end result. If you plan to replace your proprietary systems with UNIX, then you can remove any platform-specific syntax from your application completely. If you intend to supplement your existing configuration with UNIX hardware, but anticipate that further development of the system under UNIX may eventually be returned to the original platform, then conditional compile statements placed around machine-specific syntax will isolate it:

Conditional compilation enables the PowerHouse programmer to maintain and control a single program specification for multiple platforms.

Platform specific conditional compile keywords are also available for each of the UNIX platforms. UNIX is recognized by all of them.

```
quiz cc="(UNIX)"
...
> @if hpmpe
>         SET REPORT DBMODE 5
> @elseif UNIX
>         SET REPORT DEVICE PRINTER 'lp -d lp4'
> @endif
```

If your application depends on external subroutines for part of the processing, then these 3GL routines must be ported too. A C compiler is traditionally included as part of the UNIX operating system, and the obvious choice for the developer is to rewrite external subroutines in C. However, experience has shown that this often takes longer to do than anticipated, and it may be more cost-effective to look for a comparable 3GL compiler on your target UNIX platform. If you are currently using a relational database, then availability of the appropriate pre-compiler required by any 3GL programs or subroutines will also need to be investigated.

PowerHouse Conversion Aids

A couple of UNIX shell scripts are provided with the PowerHouse product to facilitate the conversion process. For a default installation of PowerHouse (where *nnn* is the version), the scripts can be found in the directory:

/usr/cognos/nnn/migrate

The following briefly describes the purpose of each script -

phlower
Downshifts all program specifications except for quoted strings and comments after a semicolon.

phfilename.csh
Alters the case of all transferred file names.

Physical Transfer

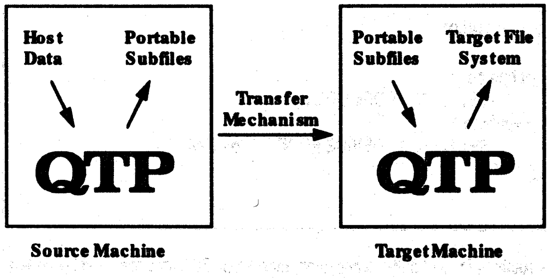
Source and data can be physically transferred using methods such as:

- Over a Local Area Network
- Using a serial line communications protocol such as Xmodem, Kermit, etc.
- Using a hardware vendor's tape-to-tape utility for source and target systems

The PC approach is more time-consuming, but is most commonly available because only a serial communications line is needed.

Where the source and target file systems are different, PowerHouse provides an independent format for transfer called Portable Subfiles. These files simply store data in ASCII format.

Data must be unloaded from the host data files to Portable Subfiles using the PowerHouse volume transaction processor (QTP) and then moved to the target machine using the chosen transfer mechanism. Once on the target UNIX system, the PowerHouse volume transaction processor can be used again to reload the data.


Host Data
Portable Subfiles
QTP
Transfer Mechanism
Portable Subfiles
Target File System
QTP
Source Machine
Target Machine

Borland's InterBase offers a transport independent data format through its backup utility gbak.

For data resident in an RDBMS, the database vendor may provide their own transport independent method of data for transfer. Alternatively, database network connectivity may be available between source and host platforms making data transfer fast and simple by using the vendor's supplied tools or utilities.

Here's how it's done in simple terms:

In QTP on MPE machine:

```
>ACCESS <filename>
>SUBFILE <subfilename> PORTABLE INCLUDE  <filename>
```

Data half "<subifilename>Q" copied in Binary and renamed to <subfilename>.ps
Dictionary half "<subfilename>" copied in Ascii and renamed to <subfilename>.psd

In QTP on UNIX machine

```
>ACCESS *<subfilename>
>output <filename> append
```

Batch Processing

One seemingly major change would be the batch processing. In fact, all that is required is to remove the job card from your stream files and simply place an ampersand after the program called.
e.g. On MPE a script called "quizrun01" might contain:

```
!JOB SOMEJOB,USER.GROUP
:QUIZ
ACCESS EMPLOYEES
CHOOSE EMPLOYEE PARM
REPORT ALL
GO
1
2

EXIT
!EOJ
```

Becomes the following on UNIX

```
QUIZ << EOF                    ;This accepts input from the file that this is in until the
ACCESS EMPLYOYEES              ;marker "EOF" is encountered.
CHOOSE EMPLOYEE PARM
REPORT ALL
GO
1
2

EXIT
EOF
```

To execute the script on MPE you would use "**stream quizrun01**, on UNIX use "**batch quizrun01**".

You'll find that this sort of thing will come easy after a while. There really isn't that much difference between the two.

Temporary Files

Since temporary files really don't exist on UNIX, you may wonder how we got around them. What the PowerHouse products do is place any "temporary" files in a directory that is composed of the programs process id. For example: If your QUIZ session has a process id of 12345, all temporary subfiles will go into a directory called *ph12345.tmp*. However, if a QUICK process is started and has a process id of 10000, and then QUIZ is called from QUICK giving QUIZ a process id of 10001, the temporary files will go into *ph10000.tmp*. **It is always the "parent" process that initially creates the temp directory.** The temp directory gets deleted when the user exits the parent PowerHouse product.

Using this method results in relatively unique files for each user. If you like, you can control where the temp files go by setting an environment variable called PHTEMP to point to the directory that the temp files should go.

Dynamic Screen Calling

With HP MPE, one can back reference file equations. This feature forces the file equations to be re-evaluated every time they are referenced. One may use syntax such as:
SUBSCREEN *somescreen
{Where there previously exists a file equation as **somescreen**}

With UNIX, this feature does not exist. Symbolic links are not evaluated every time they are referenced. However, with version 723, you will be able to call a screen that has its name stored in a temporary variable.
e.g. **SUBSCREEN ITEM somescreen**
{Where *somescreen* is a temporary that has been set to equal to a screen name}

Block Mode

Block mode is not available with PowerHouse on UNIX. Panel mode should be used instead.
Where you previously had a screen statement with the option **BLOCKMODE**, you would now change it to be **PANEL**. Panel mode processing is just like block mode except for the fact that it allows you to group fields into panels instead of being restricted to have the whole screen in one block. This feature is also available with versions of PowerHouse on MPE/iX.

Security

Since the application security classes are not compatible between UNIX and MPE, you will have to alter your dictionary to reflect security as an aspect of user id and group id instead of logonid.

e.g.
ASC MIS LOGONID *smithj.users*
would be changed to
ASCI MIS UIC [*uid,gid*] {Where uid and gid for a user can be found by using the command "*id*"}

Variable setting

On MPE, you could set a variable by either using a **RUN COMMAND** "SETVAR ..."; on UNIX, you run into the problem of subprocesses. When you do a **RUN COMMAND** to set an evironment variable, it is only valid for that subprocess and will not be set for consecutive **RUN COMMANDs**. However, with version 723 of PowerHouse for UNIX, you will be able to set variables at the level of the parent process (QUICK) with **GETSYSTEMVAL** and **SETSYSTEMVAL**.

Parameter passing

Techniques for parameter passing will also differ.

The following script runs a QUIZ report without prompting. Calling the script with "show status" 12 causes the expansion of the $1 parameter to show status and $2 to 12 in response to the parm prompt.

The "eof" is a user-defined marker string, redirecting command input to the rest of the script until the repetition of the string.

```
#! /bin/sh
quiz << eof
$1
define company_id character*2 = parm
report all
go
$2
exit
eof
```

Conclusion

Converting to UNIX can be a straightforward process when using PowerHouse. Nevertheless, schedules for conversion should include sufficient time for familiarization with the new operating system environment, the editor, establishing the physical data transfer mechanism, and for the conversion of any processing external to the PowerHouse program specifications. If the conversion includes moving to a relational database and/or a Client/Server Windows environment, additional time for planning, possible data redesign and education will also need to be considered.

Systems Management
MPE/iX BackUp Review and Implementing New Technology Overview

by Al Dulaney

Hewlett-Packard systems are used primarily for commercial on-line transaction processing (OLTP). Increasingly, these OLTP applications are becoming operationally critical--if the application stops. Users of such applications need systems that are protected from both hardware and system software failures.
HP 3000s have several standard features that enhance the reliability of hardware and system software:

- Highly reliable PA-RISC (Precision Architecture-Reduced Instruction Set Computing) based processors.
- Capability of the processor hardware to continue processing after temporary power failures without incurring processor failure or losing data.
- An integrated transaction management facility that automatically checkpoints and logs activities for critical system data structures, user files, and databases. This facility guarantees data integrity in the event of a system failure.

The objective then, is to identify and recommend solutions that address the process, people and technology of today's current environment. Augment these standard features with a comprehensive backup and recovery strategy that needs to be planned, tested and implemented.

Issues

The challenges for Information Technology (IT) are to:

- Improve store management that encompasses backup and restore, disk management, tape management, and database management.
- Improving system availability by eliminating downtime due to data backup, the primary cause of planned downtime.
- Reducing operating staffing requirements and cost.

IT's desired result is the management of expenses and improving its responsiveness to the changing business needs of the corporation and the requirements of its users.

Solutions are not merely hardware and software. Technology alone is not sufficient to meet business objectives. Technology is merely one of the components that allow your people to automate the correct and repeatable processes that run your business. The SSO, through its team of knowledgeable people and successful project experiences can help guide your staff in refocusing processes, selecting, implementing, and deploying technology for a reliable, manageable production environment.

Discussion

Over the next hour I'd like to discuss the implementation of new hardware and software technology in a production environment.

- Document and review your current backup and recovery procedures.
- Identify targeted backup time windows in the production schedule
- Document your current hardware configuration by producing a worksheet of your current system configuration resulting in identifying any potential hardware performance bottle necks.
- Review disk management practices for the use of volume management techniques.
- Review backup media and hardware options (i.e., DAT, 7980, 7980XC, Optical).
- Recommend any configuration changes to take full advantage of new software, and hardware technologies.
- Recommend workload modification to accommodate backup window.
- Develop testing methodology and procedures using the selected solution.

First, let me give you some background about the customer situation. The customer informed me that they were getting some new hardware, and software and that they had lost one of their system managers. They were in the process of hiring a new person but, in the interim they ask me to work with them as a system manager and implement the new technology.

First, we defined the objectives:

- Identify what hardware and software they had purchased
- Identify the current operating system environment- Identify which release of the operating system
- Identify whether any patches were required for this new hardware and software
- Identify the available resources, both internal to HP and customer
- Develop a plan to implement the identified products

The desired result was to have a smooth implementation while moving to the new hardware and software technologies into the production environment. In addition, to have a limited amount of down time with no lost time due to misconfigurations, or bad software patches and no loss in performance. Performance is a key issue here. We needed to monitor each step to make sure that we were not loosing any performance or functionality, as we implemented the new hardware and software technology.

The customer's environment is an 980-100 production and 925LX development HP 3000. The implementation of the disk arrays and TurboSTORE/iX II with On-Line backup could be done simultaneously.

1) Implementation of volume management on disk arrays

The customer purchased one 5.4 disk arrays for high availability of their Lab Management System (LMS). We discussed which accounts should be placed on the user volume set.

We identified the customer's operating system to be B.03.00. B.04.00 had just come out but not all of our customers, especially our production customers, had gone to 4.0. Patches for 3.0 disk arrays and TurboSTORE/iX II were needed. In addition, we applied the latest 3.0 power patch tape. We had the CE do a site-prep for the hardware installation. We wanted to make sure there was enough floor space for the disk array cabinet and power. The delivery of the hardware was confirmed and scheduling of the installation was planned.

The disk array was put in place and powered on as soon as it arrived. The system would have to be shutdown to install the fiber-link (FL) card. We reviewed the SYSGEN configuration parameters. This particular customer had the option of being able to add the disk array without having to take any existing disk drives off. Another option would be to replace or reassign existing disk drives which would involve an INSTALL. Worst case, we would have to plan for a complete weekend for the INSTALL and reconfiguration. As it turns out, we were able to implement our plan over a couple of weekends without an INSTALL. The hardware was installed one weekend with the appropriate configuration changes. The SYSGEN changes were made in advance and placed in a separate "config" group.

We used BULDACCT.PUB.SYS to create the job streams for the accounts identified for the user volume set. We discussed the naming conventions and planned which accounts were to go in the user volume set. The next issue was how to store the files to tape and restore then on the user volume set as quickly as possible. The customer estimated that it would take 12 hours to store the production account to tape and 12 hours to restore it onto the user volume set. We had to plan for a large window of time in order to do this. The windows we had were only on the weekends since there were no three day weekends in the near future . Our approach was to look at the accounting structure for that account, and look at the number of groups. We quickly determined that one group was half the size of the entire account and the remaining groups were half the size. The customer has two 6250 tape drives, so our approach was to store the main group to one tape drive and the remaining groups to the second tape drive. The 12 hour store could be cut down to 6 hours using both drives.

We would then initialize the disk arrays and stream the BULDACCT job streams to create the user volume set. We planned for two simultaneous stores and restores. Instead of looking at 24 hours, we're looking at 12 hours. The system would still be up at that time, but the user for this particular application would be unavailable.

One other issue was the fact that the users who create files may or may not be on the system. If you store files within the MPE/iX command and do a restore, you get a nonexistent user who is not in the accounting structure. You have to go back and identify those files, groups, and users, then do a restore with a creator equals MGR or what ever name you may choose. We didn't want to do that because that would take files that hadn't been accessed for a long period of time and force them on the system with an active user name. Our other approach was to use VESOFT's MPEX utility. It has a function for looking for nonexistent users. John, the system manager, very quickly sat down and ran this utility and entered the command. He ran it against the account and he found that there were several users or creator of files that were no longer on the system. We also noticed the access date was prior to 1992 and John quickly purged these files. I questioned his action. Why did he purge those files? They could be important production files? He said they have a yearly backup of those files and they hadn't been accessed for over a year. If anybody needs them, we could restore from that backup tape.

John identified those files that were being accessed and then used the MPEX command to change their creator to MGR. This enabled us to eliminate a second restore. We have cut down our store and restore to the bear minimum of 12 hours. We could test the BULDACCT job streams while the system was up because we were able to put the hardware on and we didn't have to replace any system domain disk drive.

One other important step was to identify the current hardware configuration. To accomplish this I ran SYSDIAG with SYSMAP function and printed off the I/O configuration of the 980-100 and then used an editor to draw this logical map of existing hardware. *See Appendix A-1.*

The main concern here was, whether we had enough slots available. I was concerned for the performance, whether we had too many of one type disk drive on an HPIB or FL card. The disk array was going to be configured on its fiber link card, and we had to have an open slot. An additional Channel I/O Bus (CIB) was purchased to handle the I/O expansion. The DDS tape drives were HPIB and we had to see if we had additional HPIB card to put these on. DDS tape drives are very slow, the SCSI drives were just coming out and the data compression SCSIs were not yet available. My main concern was that I put all four of these drives on one HPIB card and it would be extremely slow in comparison to 7980 and 7978.

I modified the first worksheet to plan the proposed changes. We would plan which path, device number, and device class we would use. *See Appendix A-2.*

The placement of the four (4) DDS dat tape drives was also a concern. For performance we placed two (2) DDS dat tape units on one HPIB card and the remaining two (2) on another with the two (2) 6250s drives. They would not be used at the same time for backups. This was done because the C2254HA disk arrays were new and the expected performance was not yet known. We made a contingency configuration for mirrored disk. We thought about different methods of laying out the existing disk drives to implement mirrored disk. This was planned because, if the disk array performance was not up to expectations, disk mirroring would be implemented.

Why the contingency plan? The LMS was the main production application and it would be completely on C2254HA one disk array. As a backup then, we needed a contingency plan to go to mirrored disk. We had ample 7937s fiber link drives to put up a mirrored disk configuration. It would be quite a bit of effort to configure, initialize and take the system down to do that, but we still needed to put that contingency plan in place.

Planning and more planning was needed to eliminate as many possible areas of concern as we could humanly eliminate. *See Appendix A-3. We* followed the user volume creation checklist and used the BULDACCT utility. *See Appendix A-4 and 5.*

2) Implementation of TurboSTORE/iX II with On-Line Backup

Implementation of TurboSTORE/iX II with On-Line Backup with DDS tape drives presented a different set of problems. First of all, we had to review the current backup procedures to find out exactly what they were doing. As it turned out, they were using SILHOUETTE/3000 to shadow their databases from the 980-100 across to a Series 70 and then backing up the databases on the remote system. They informed me that they were taking the Series 70 out and that database backup/recovery would now be shifted to the 980-100 production machine. Current backup procedures had to be reviewed and modified to accommodate these changes.

In addition, they wanted to review their current TurboIMAGE logging procedures and see if they could take advantage of the features of 3.0: Roll-Forward, Roll-Back and Dynamic Roll-Back. We looked at the current backup streams, and the current backup schedule. We ordered the appropriate software add-on tape that was purchased and that was updated to System-E, which is a development system. We were able to test some of the backup options: interleave, sequential, parallel, compression low, compression high and different combinations on the development system. The development system already had user volume sets. In the past all you had to do was get a compression tape drive or a bigger tape but you didn't have the variety of methods you do today.

Placement of the HPIB drives was a major concern. Again we went back to the worksheet to make sure that the drives were placed on different channels, because HPIB is a 1 megabyte bandwidth. We didn't want to saturate that bandwidth, because we had no tools, no method as to test to see if we are having adequate throughput. Also, the customer wanted to do parallel backups and the HPIB cards have to be on different CIO Buses. We put units 34/35 on one HPIB card and units 36/37 on another. That way we could use even numbers, 34/36 to do one set of user volume backups and 35/37 to do the others.

We wanted to test simultaneous on-line backups to two different volume sets. We experienced no problem with shadow files or simultaneous on-line backups. We could do two different backups simultaneously or whatever the schedule permitted. We did this on the development machine in lieu of testing it on the production machine.

One of the things I quickly found, I didn't have a database that I could start and stop in order to test out the logging options. They requested a review of the new database logging features set that is now available TurboIMAGE. In order to do this, I looked around to see if I could find a sample database and an update program that would allow me to test the new features, and the different recovery methods. I could then stop it or break database followed by the test of recovery procedures.

In the TurboIMAGE manual, there is a COBOL example that does use all of the TurboIMAGE intrinsics. I could use those files for roll forward and roll backwards test. I went into CD-ROM, extracted the data to a flat file, linked it down to my portable and then to the system. I edited out the text resulting in a SCHEMA and a COBOL update program. All I had to do then was create some dummy data using QUERY to add the different datasets. I created several job streams that would add the masters, details, and create the automatic masters. Additional QUERY job streams to go through and erase the data sets, so I could reproduce them again.

With little or no effort I was able to have a testing database. This represented their production logging environment as I viewed it. We reviewed the procedures of stopping the old logging file, switch logging over, starting the log again, and resuming with on-line backup. These procedures have been worked out on paper, then we tested them on the test database.

We did not use DBSTORE to put a time stamp in the database. That would have us use one tape per database and one per log file. We used TurboSTORE/iX II to backup the logging files and databases with no time stamp. The coordination of this becomes very important that we keep the tapes and log files labeled properly. We need to know which tapes contain both the database and logging files. During the recovery, we get the log files off that tape along with the database. In the past, we had different tapes to store the database, and the log file. This method does cause us concern, that the dat tape drive holds both the TurboIMAGE log file and the database. We need to watch this and make sure the tape drive gets cleaned on a daily basis and that tapes after 200 passes gets recycled. The customer requested this method be tested to reduce the total number of tapes and tape management. Our implementation resulted in fewer DBSTOREs, fewer tape mounts by the operator, fewer tapes to be labeled, fewer tapes to be entered into a tape management system, and less storage spaces for tapes.

We tested different backup combinations. We first tested with compress = low, then compress = high. We found that the 980-100 had enough horse power to use the compression high and also to use parallel. Because we had aligned the DDS tape drives, two on one HPIB card and two on another, we could do two simultaneously on-line backups stores in parallel. We need two drives available to complete restore, so we have some more operating procedures to develop than we had in the past. In the past, you just mounted a tape for the store, or sequential store. You had to make sure the operator labeled the tapes and if it was 20 or 30 tapes, make sure he labeled them in correct sequence. Now, we're going to use these DAT drives, there are four in the cabinet, two go to one store, two go to another. Now labeling conventions become ever more important. The customer chose not to use label tapes, but label tapes would be one way to correct this. Mainly the DDS tapes are for backups, so regular stores, during the course of the week, are not done by the 6250s. These labeling procedures should be revisited and updated to make sure these tapes are label properly.

Test	Turbo STORE Options	Hours	Mins.
1	COMPRESS=LOW	3	180
2	COMPRESS=HIGH	2.17	137
3	COMPRESS=HIGH;PARALLEL	1.21	87

This all resulted in higher availability of the system. Mainly because we used on-line backup and this is a continuing manufacturing process They stop the process at midnight and then they bring the system users down and within 10 minutes they can get back on the system. We really truly are a 23X50 hours per day and seven days a week operation now, because the customer is only down for 5 or 10 minutes. We were able to reduce the amount of time the full backup took. The wall clock time was not a major concern as shown by the table above. The slowness of the tape drive is overshadowed by the fact that we can get more people on, stay on longer and we can still get the tapes through there by using the compression options=high, and parallel.

The things that I can't test are the volume load. It is inconceivable to have a test development system with same amount of volume as the production system. We can validate the process and job streams on the development system. Once I tested that, I feel very comfortable that when we put it in the production environment. The only thing I'm going to have in addition to that is the amount of volume, amount of free space we need for the shadow files on each volume set and the number of tapes required. We can review the load leveling situation, where I might switch some accounts around and store at separate, to balance the file distribution.

3) Implementation plan in production environment

Implementation of the plan is as critical as doing the test, because you have to find the time to stop the system, add the hardware, software, and patches to the production system. We have to plan out the user environment, and volume management changes. Make the changes to SYSGEN, then schedule one weekend to do the store and restores. Once these steps were completed on the development system, we then moved them over to the production system. *See Appendix A-6.*

After implementation the backup process still has to be revisited on a periodic basis, because things change. TurboSTORE statistics should keep and monitor the number of tapes that are used, the length of times, and any $STDLIST messages. We still want to make sure that when all our data is backuped we want to make sure we can restore from that tape. Additional test should be carried out in your environment to identify what could be a possible weakness and tighten the process up.

Today, that environment has changed. They have now moved disk arrays into ldev 1 and ldev 2. They have switched from 2254s to 2252s. We would still use disk arrays for high availability and user volume management. We're still using DDS dats HPIB for backup. The customer is pleased with the new technology. The new technology we could implement today:

- New SCSI DDS tape drives with two (2) gigabits of storage capacity
- Data compression SCSI DDS tape drives
- Larger capacity 2 Gbyte disk arrays, C2258HA/B and C2259HA/B
- Rewritable Optical Disk Library System

Summary

Document current backup and recover procedures. Identify the targeted backup and time window for instruction schedule. This particular example was a continuing manufacturing process, where everyone could be gotten off the system at a prescribed time. In your environment, that might not be the case, but by locating accounts on separate volume sets, that volume set might be closed a 8:00 or backuped at 8:00 while the others stay up. User volume sets could be used and are an integral part in the developing of your backup strategy because you can group users on volume sets that could be backed up at certain times. Payroll, finance could be backed up possibly any time, your manufacturing you want to group on a separate volume set. So volume sets play a very important role in your backup recovery procedures. So that's how I group users to fit into your schedule for backup procedures.

I can't say enough about planning, documenting your current hardware configuration and making these worksheets. A lot of time and effort go into these worksheets, but I can sit down with you the customer, to review options or plan different scenarios. We can then sit down with the CEs to determine if this is viable solution. We can discuss this with the performance specialists to make sure we don't have too many drives on one channel. We can use this as a worksheet when we do the configuration and SYSGEN. All this is an effort to minimize the amount of down time and minimize the number of problems we are going to run into when we do the switchover. We can't defense each and every situation. The best we can do is review each and every step, do it in the daylight hours, so when we get in there to do backups and to move discs and tapes around late at night, we will have a clear understanding of the I/O paths or sequence of events. Volume management is critical to this implementation and goes hand in hand with your backup recovery. Along with that is a series of commands that you need to review some command files that help you build user volume sets, manage your volume sets and rebuild your volume sets in the case of a disk problem. The recovery procedures for one of those modules need to be reviewed and tested.

We worked together as a team. We accepted the challenge of implementing the new technology that is facing the Information Technology (IT) department today. We improved the store management process that encompasses backup, restore, disk management, tape management, and database management. We improved system availability by eliminating downtime due to data backup, the primary cause of planned downtime. We reduced the operating staff's requirements concerning the backup process, the associated cost of tapes and their storage. There was no question that the hardware and software would work. The only question we had, could we implement the process we had planned in a timely and save manner in a production environment. This paper is a testimony to the team's success.

```
                  (C U R R E N T)
                      980-100
    +-------------------+ +-------------------+
    |Central Bus (CTB) 0| |Central Bus (CTB) 1|
    +-------------------+ +-------------------+
      2/4         2/8         6/4         6/8
  +---------+ +---------+ +---------+ +---------+
  | CIO BUS | | CIO BUS | | CIO BUS | | CIO BUS |
  |(CIB) 0_1| |(CIB) 0 2| |(CIB) 1_1| |(CIB) 1 2|
  |         | |Optional | |         | |Optional |
  +---------+ +---------+ +---------+ +---------+
   0 1 2 3 4   0 1 2 3 4   0 1 2 3 4   0 1 2 3 4
   H M L H A               O D O H H
   P U A P P               P T P P P
   I X N F |               E C E I F
   B | | L |               N | N B L
   | + 3 1 +                 +   | 2
   +                       _   _ +
 /Switch/                      /Switch/
 +-----+                       +-----+ +-----+
 |2680A|                       |7980A+++7978B|
 +-----+                       +-----+ +-----+
   HP-FL        HP-FL
+---1----+   +---2----+   +---3---+
|7937FL  |   |7937FL  |   | S/70  |
|Master1 |   |Member4 |   +-------+
|LDEV 1  |   |LDEV 4  |SILHOUETTE/3000
|disc,   |   |disc,   |
|spool   |   |spool   |
+--------+   +--------+
|7937FL  |   |7937FL  |
|Member2 |   |Member5 |
|LDEV 2  |   |LDEV 5  |
|disc,   |   |disc,   |
|spool   |   |spool   |
+--------+   +--------+
|7937FL  |   |7937FL  |
|Member3 |   |Member8 |
|LDEV 3  |   |LDEV 8  |
|disc,   |   |disc,   |
|spool   |   |spool   |
+--------+   +--------+
|7937FL  |   |7937FL  |
|Member12|   |Member9 |
|LDEV 12 |   |LDEV 9  |
|disc,   |   |disc,   |
|spool   |   |spool   |
+--------+   +--------+
|7937FL  |   |7937FL  |
|Member14|   |Member11|
|LDEV 14 |   |LEV 11  |
|disc,   |   |disc,   |
|spool   |   |spool   |
+--------+   +--------+
|7937FL  |
|Member15|
|LDEV 15 |
|disc,   |
|spool   |
+--------+
                   (P H A S E  I)
                980-100 224mb Memory
      +-------------------+ +-------------------+
      |Central Bus (CTB) 0| |Central Bus (CTB) 1|
      +-------------------+ +-------------------+
    +---2/4---+ +---2/8---+ +---6/4---+ +---6/8---+
    | CIO BUS | | CIO BUS | | CIO BUS | | CIO BUS |
    |(CIB) 0_1| |(CIB) 0_2| |(CIB) 1_1| |(CIB) 1_2|
    |         | |         | |         | |Optional |
    +0-1-2-3-4+ +0-1-2-3-4+ +0-1-2-3-4+ +0-1-2-3-4+
     H M L H A   H H O O O   O D O H H
     P U A P P   P P P P P   P T P P P
     I X N F |   F I E E E   E C E I F
     B | | L |   L B N N N   N | N B L
     5 + + 1 +   3 4 _ _ _   _ + _ 6 2

  HP-FL      HP-FL      HP-FL               /Switch/
 SYSVOLSET  SYSVOLSET    FDT       HP-IB     HP-IB
+---1----+ +---2----+ +---3----+ +---4---+ +---6---+
|2/4.3.0 | |6/4.4.1 | |2/8.0.0 | |2/8.1.0| |6/4.3.1|
|7937FL  | |7937FL  | |C2254HA | | DDS03 | | 7980A |
|Master1 | |Member4 | |Master1 | |LDEV 36| |LDEV  7|
|LDEV  1 | |LDEV  4 | |LDEV 80 | |tape   | |tape   |
|disc,   | |disc,   | |disc    | +---+---+ +---+---+
|spool   | |spool   | |        |     |         |
+--------+ +--------+ +--------+ +---+---+ +---+---+
|2/4.3.1 | |6/4.4.2 | |        | |2/8.1.1| |6/4.3.5|
|7937FL  | |7937FL  | |        | | DDS04 | | 7978A |
|Member2 | |Member5 | |        | |LDEV 37| |LDEV 33|
|LDEV  2 | |LDEV  5 | |        | |tape   | |tape   |
|disc,   | |disc,   | |        | +-------+ +---+---+
|spool   | |spool   | |        |               |
+--------+ +--------+ +--------+ +-------+ +---+---+
|2/4.3.2 | |6/4.4.3 | |2/8.0.1 | |2/12   | |6/4.3.6|
|7937FL  | |7937FL  | |C2252HA | |  PSI  | | DDS02 |
|Member3 | |Member6 | |Member1 | |LDEV 17| |LDEV 34|
|LDEV  3 | |LDEV  8 | |LDEV  81| |kc     | |tape   |
|disc,   | |disc,   | |disc    | +-------+ +---+---+
|spool   | |spool   | |        |               |
+--------+ +--------+ +--------+ +-------+ +---+---+
|2/4.3.3 | |6/4.4.4 |            |6/12   | |6/4.3.4|
|7937FL  | |7937FL  |            |  PSI  | | DDS03 |
|Member9 | |Member7 |            |LDEV 18| |LDEV 35|
|LDEV 12 | |LDEV  9 |            |ca     | |tape   |
|disc,   | |disc,   |            +-------+ +-------+
|spool   | |spool   |
+--------+ +--------+            +---5---+
|2/4.3.4 | |6/4.4.5 |            |2/4.0.7|
|7937FL  | |7937FL  |            | 2680A |
|Member10| |Member8 |            |LDEV 16|
|LDEV 14 | |LEV 11  |            |laser  |
|disc,   | |disc,   |            +-------+
|spool   | |spool   |
+--------+ +--------+
|2/4.3.5 |
|7937fl  |
|Member11|
|LDEV 15 |
|disc,   |
|spool   |
+--------+
                  (Proposed P H A S E  II)
                        Mirrored Disk
                           980-100
         +--------------------+ +--------------------+
         |Central Bus  (CTB) 0| |Central Bus  (CTB) 1|
         +--------------------+ +--------------------+
       +---2/4---+ +---2/8---+ +---6/4---+ +---6/8---+
       | CIO BUS | | CIO BUS | | CIO BUS | | CIO BUS |
       |(CIB) 0_1| |(CIB) 0_2| |(CIB) 1_1| |(CIB) 1_2|
       |         | |         | |         | |Optional |
       +0-1-2-3-4+ +0-1-2-3-4+ +0-1-2-3-4+ +0-1-2-3-4+
        H M L H A   H H O O O   O D O H H
        P U A P P   P P P P P   P T P P P
        I X N F |   F I E E E   E C E I F
        B | | L |   L B N N N   N | N B L
        | + + 1 +   3 4 _ _ _   _ + _ | 2
        |                             |
/Switch/+                          /Switch/
+-------+                             \
|2/4.0.7|                              \
| 2680A |                               \
|LDEV 16|                                \
|laser  |                                 \
+-------+                                  \
   HP-FL      HP-FL      HP-IB      HP-FL    +---+
  Mirror_1   Mirror_2   SYSVOLSET    FDT        HP-IB
 +---1--=-+ +---2--=-+ +---7----+  +---4---+ +---+---+
 |2/4.3.0 | |6/4.4.0 | |2/8.3.0 |  |2/8.1.0| |6/4.3.1|
 |7937FL  | |7937FL  | |7937XP  |  | DDS04 | | 7980B |
 |Master1 | |Master1 | |Master1 |  |LDEV 36| |LDEV  7|
 |LDEV 40 | |LDEV 60 | |LDEV   1|  |tape   | |tape   |
 |disc    | |disc    | |disc    |  +---+---+ +---+---+
 |        | |        | |        |                 |
 +--------+ +--------+ +--------+  +---+---+ +---+---+
 |2/4.3.1 | |6/4.4.1 | |2/8.3.1 |  |2/8.1.1| |6/4.3.5|
 |7937FL  | |7937FL  | |7937XP  |  | DDS05 | | 7078B |
 |Member2 | |Member2 | |Member2 |  |LDEV 37| |LDEV 33|
 |LDEV 41 | |LDEV 61 | |LDEV   2|  |tape   | |tape   |
 |disc    | |disc    | |disc,   |  +-------+ +---+----+
 |        | |        | |spool   |                 |
 +--------+ +--------+ +--------+  +---5---+ +---+---+
 |2/4.3.2 | |6/4.4.2 | |2/8.3.2 |  |2/12   | |6/4.3.6|
 |7937FL  | |7937FL  | |7937XP  |  |  PSI  | | DDS02 |
 |Member3 | |Member3 | |Member3 |  |LDEV 17| |LDEV 34|
 |LDEV 42 | |LDEV 62 | |LDEV   3|  |kc     | |tape   |
 |disc    | |disc    | |disc,   |  +-------+ +---+---+
 |        | |        | |spool   |                 |
 +--------+ +--------+ +--------+  +---6---+ +---+---+
 |2/4.3.3 | |6/4.4.3 | |2/8.3.3 |  |6/12   | |6/4.3.4|
 |7937FL  | |7937FL  | |7937XP  |  |  PSI  | | DDS03 |
 |Member4 | |Member4 | |Member4 |  |LDEV 18| |LDEV 35|
 |LDEV 43 | |LDEV 63 | |LDEV   4|  |ca     | |tape   |
 |disc    | |disc    | |disc,   |  +-------+ +-------+
 |        | |        | |spool   |     FDT
 +--------+ +--------+ +--------+  +---3---+
 |2/4.3.4 | |6/4.4.4 | :2/8.3.4 :  |2/8.0.0|
 |7937FL  | |7937FL  | :7937    :  |C2254HA|
 |Member5 | |Member5 | :Member5 :  |Master1|
 |LDEV 44 | |LEV 64  | :LDEV   5:  |LDEV 80|
 |disc    | |disc    | :disc,   :  |disc   |
 |        | |        | |spool   |  +-------+
 +--------+ +--------+ +........+
```

USER VOLUME MANAGEMENT
Volume Set Creation/Migration Check List

Procedures for moving from one MPEXL_SYSTEM_VOLUME_SET system domain, to multiple volume sets:

1.) Create a newgroup in the SYS account to hold the BULDACCT jobstreams (ie. BULDACCT). The two jobstreams created by BULDACCT (BULDJOB1, and BULDJOB2) will assist in creating the necessary accounting structure to accommodate your new system environment.

2.) CHGROUP to the new group you wish to contain the BULDACCT jobs.

3.) Determine which accounts will be moved to which User Volume Sets. Run BULDACCT.PUB.SYS with the appropriate options (see attached output listing of BULDACCT's options) to move your accounts or groups to a specified Volume Set.

For example: To just move the MFG account to MFG_SET1, use the following BULDACCT run statement:

```
:RUN BULDACCT.PUB.SYS;INFO="MFG%VSACCT=MFG_SET1"
```

the resulting BULDJOB1 job will contain the necessary command structure to rebuild the MFG account and groups in both the system directory and the user set directory of the MFG_SET1 volume set.

NOTE: At this point, these user volume sets do not exist. However, these tasks can be performed ahead of time to help expedite the migration once the actual migration is under way - minimizes system down time.

4.) If you are setting up more than one volume set, you may wish to rename BULDJOB1 and BULDJOB2 to an appropriate name (ie. BULDMFG1, BULDMFG2), and repeat step '3' for another set or accounts and volume set. Failure to rename, and the BULDJOBxs will be overlaid with your next account.

5.) Assign 'UV' (user Volume) capability to all users expecting to access a User Volume set. This capability is required to enable users to access a User Volume set and can be assigned ahead of time using the 'ALTUSER' command or adding UV cap option to the BULDJOB1 file with editor of your choice.

6.) Modify the SYSSTART.PUB.SYS file to contain the following command:

```
VMOUNT ON,Auto
```

Use the 'Delete Volume' command (DV) to remove all drives that you do not wish to be part of the new system domain's MPEXL_SYSTEM_VOLUME_SET.

Once this is completed, save the changes (type HOLD)and Exit out of the 'IO' module. Keep the new configuration into the CONFIG.SYS group (KEEP).

7.) Run SYSGEN and enter the 'IO' module (type IO). Perform a 'List Volume' command (LV), to determine which volumes are designated to be automatically initialized as system drives during an INSTALL (thus eliminating the need to use VOLUTIL to add these drives to the system domain before that can used).

8.) Create a new System Load Tape (SLT) from SYSGEN using the 'TAPE' command. Label this tape with appropriate heading, date, system name, and drive created. Store where all operations can find it if need be. This step can be done with users accessing the system.

The following steps should be taken at the time of migration

9.) Perform a FULL backup of your system. Be sure to store the group containing the BULDACCT jobstreamsfirst, followed by @.@.@ (or your standard fileset);don't forget to use the DIRECTORY option. It would be advisable to get a hardcopy listing of your backup's $STDLIST as well. An example of the STORE command would be:

```
:file T;dev=tape
:file L;dev=lp
:STORE
@.BULDACCT.SYS,@.@.SYS,@.@.@;*T;&
:directory;offline=*L
```

IMPORTANT - Do not continue this process until you have verified that ALL files have been successfully STOREd to tape. The next step (Step 10 will erase all of your data from the disks. Confirm that the 'DIRECTORY' option was used as well.

10.) Perform an INSTALL using the SLT created in Step 8.

11.) Perform 'DSTAT ALL' command to ensure that only the drives that you are expecting to show up under the MPEXL_SYSTEM_VOLUME_SET are in the system volume set. If not, return to Step 5 to determine why.Otherwise proceed to next step.

12.) Run VOLUTIL to create the desired User Volume Sets (using the SCRATCHVOL, NEWSET, and NEWVOL commands where necessary). You should already know which drives you wish to be members of a volume set - of these drives, one disk will be the master (created using the NEWSET command). All subsequent membersare added to the newly created set using the NEWVOL command. (see 32650-90045 Volume Management Reference Manual).

13.) Once you have completed the initialization step, you should verify your work with the 'DSTAT ALL' command. If all looks good, proceed with next step.

14.) RESTORE the BULDACCT jobstreams and DIRECTORY, ie:

```
:file T;dev=tape
:RESTORE *T;@.buldacct.sys;directory;show
```

15.) Stream all of the BULDxxx1.BULDACCT.SYS jobstreams. Confirm, via the $STDLIST files, that all jobs completed successfully. Check your work with the REPORT, LISTACCT, and LISTGROUP commands; perform REPORT using the 'ONVS=' option, to endure that the groups are on the volume sets where that belong.

= Important Notice =

In a production environment, account users are added and deleted over time. So, files can exist in the account that were created by a user who no longer exist on the system. At restore time a message will displayed that the user no longer exist and the files are not restored. Steps need to be taken to guard against this happening or an addition restore must take place to get all the files restored back to the system. MPEX has a option that checks all files for all file creators who no longer exist. The MPE RESTORE command has an option to set the creator name. You have to make the decision whether to purge these files or change the creator to an existing user name. Otherwise valuable time will be taken to perform a second restore to add the additional files.

16.) If all looks to be in order, complete the migration with a complete reload of the system using the FULL backup tapeset.

```
:file T;dev=tape
:RESTORE *T;@.@.@;KEEP;SHOW;OLDDATE
```

IMPORTANT - DO NOT USE THE 'DIRECTORY' OPTION WITH THIS RESTORE.

17.) Stream all of the BULDxxx2.BULDACCT.SYS jobstreams. Confirm, via the $STDLIST files, that all jobs completed successfully. This step sets the individual UDC files to the system, account and user.

18.) *!!!!!!!!!!!! CONGRATULATIONS !!!!!!!!!!!!!!* You have successfully migrated your system environment to a combined system and user volume set environment. At this point you are ready to enjoy greater 'peace of mind' knowing that you have succeeded in dramatically reducing your exposure to downtime in the event of a disk crash (DISASTER) requiring an INSTALL to recover.

BUILD ACCOUNT PROGRAM FOR DIRECTORY MIGRATION

For use on MPE/XL systems only

Program re-builds or moves MPE/XL system accounting structure from one system to another or from one volume set to another

Account, group, user and udc accounting information is formated and written to two files which execute in batch mode (job stream). Both job stream files will execute on either an MPE V or MPE XL. The program, however,will only execute on a MPE XL system.

If no additional option is given then job streams named BULDJOB1 and BULDJOB2 are created which contain commands to recreate the directory structure on another system, identical to a snap shot of the current directory structure. BULDJOB1 recreates the accounts/groups/users and BULDJOB2 sets the system/account/user level UDCs. The following command makes the job files to recreate directory structure of all accounts:

```
:RUN BULDACCT; INFO="@"
```

Several options can be used when executing this program.Options are passed in the info=".." clause of the run command or on the interactive BULDACCT prompt which comes up when the utility is run with no info string. These options are:

1.) acct_list - A subset of accounts can be passed.This enables the user to re created only those accounts selected in account list. For example:

```
info="A@,ba@,z@"
```

2.) %VSACCT - Creates job streams with volume set option at the account level

Syntax:

```
run buldacct;info="[acct_list]%VSACCT=user_vol_set"
```

where acct_list is the list of accounts to be migrated. It can include valid CI wild carding, for example, S@, @S will include all accounts begining or ending with an S. The default acct_list is @. user_vol_set is a valid volume set name

This option causes additional newacct and newgroup commands with onvs=user_vol_set to be inserted such that the accouts and groups are migrated to the user_vol_set. As an example, if GYPSY is an account on some volume set (could be system volume set) then the following sequence of commands move GYPSY to a user volume set called "target_vol_set":

```
:STORE @.@.GYPSY ;; SHOW
:RUN BULDACCT; INFO="GYPSY%VSACCT=target_vol_set"
:PURGEACCT GYPSY {should be no system udc files in GYPSY}
:PURGEACCT GYPSY; ONVS=target_vol_set
:STREAM BULDJOB1 {rebuilds acct/grp/user structures}
:RESTORE ; @.@.@; SHOW; CREATE=CREATOR
:STREAM BULDJOB2 {resets the system/acct/user udcs}
```

3.) %VS - Creates job streams with volume set option at group level

Syntax:

```
run buldacct;info="[acct_list]%VS=user_vol_set"
```

This option causes additional only newgroup commands with option onvs=user_vol_set to be inserted such that the groups are migrated to the user_vol_set, provided the parent account already exists on the user set. As an example, if GYPSY is an account on some volume set (could be the system volume set) then the following sequence of commands move the groups in the GYPSY account to a user volume set called "target_vol_set":

```
:STORE @.@.GYPSY ;; SHOW
:RUN BULDACCT; INFO="GYPSY%VS=target_vol_set"
:PURGEACCT GYPSY
:STREAM BULDJOB1 {rebuilds acct/grp/user structures}
:RESTORE ; @.@.@; SHOW; CREATE=CREATOR
:STREAM BULDJOB2 {resets the system/acct/user udcs}
```

GYPSY account is assumed to exist on the "target_vol_set":

4.) %UV - Creates a job streams with volume set option for accounts which have atleast one group residing on a UV.

Syntax:

```
run buldacct;info="[acct_list]%UV[=user_vol_set]"
```

This option causes the account list to be shortened to those accounts, which have at least one group on a user set. If user_vol_set is specified then the additional newgroup and newacct commands with "onvs=" use user_vol_set as the target volume set. As an example, if GYPSY1 is an account on some volume set (could be system volume set) with at least one group on a user volume set and GYPSY2 has all groups on system volume set then

```
:RUN BULDACCT; INFO="GYPSY1,GYPSY2%UV"
```

creates jobs to recreate directory structure of GYPSY1 only. And

```
:RUN BULDACCT; INFO="GYPSY1,GYPSY2%UV=user_vol_set"
```

creates jobs to recreate directory structure of GYPSY1 only with user_vol_set as the volume set name in "onvs=" clause of the groups which actually are on a user (private) vol set.

5.) %HELP - causes this information to be printed.

Program Options (use these in "INFO="or BULDACCT prompt below)	
"%HELP"	to get detailed help on options
"%QUIT"	quit from interactive prompt
"acct_list%VSACCT=user_set"	to migrate accounts and groups to the user_set volume set
"acct_list%VS=user_set"	to migrate only the groups to the user_set volume set
"acct_list%US[=user_set]"	to select accounts with at least one group on the user_set

```
(Maximum info string length allowed is 80 chars)

Type the info string (max 80 chars, CR for default of @)

                    BULDACCT: ITF3000

Groups with PRIV VOL=YES will have HOMEVS= and ONVS=options.
ACCOUNT ITF3000 ADDED TO FILE

Please check that the manager.sys passwords on the job card of Buldjob1 and
Buldjob2 are consistent with the target system

    build account JOB "BULDJOB1" built..
    Setcatalog JOB "BULDJOB2" built..

END OF PROGRAM
```

```
               (Current P H A S E  III)
                        980-200
      +-------------------+ +-------------------+
      |Central Bus (CTB) 0| |Central Bus (CTB) 1|
      +-------------------+ +-------------------+
    +---2/4---+ +---2/8---+ +---6/4---+ +---6/8---+
    | CIO BUS | | CIO BUS | | CIO BUS | | CIO BUS |
    |(CIB) 0_1| |(CIB) 0_2| |(CIB) 1_1| |(CIB) 1_2|
    |         | |         | |         | |Optional |
    +0-1-2-3-4+ +0-1-2-3-4+ +0-1-2-3-4+ +0-1-2-3-4+
     H M L H A   H H O O O   O D O H H
     P U A P P   P P P P P   P T P P P
     I X N F |   F I E E E   E C E I F
     B | | L |   L B N N N   N | N B L
     6 + + 1 +   3 4 _ _ _   _ + _ 5 2

  HP-FL      HP-FL      HP-FL              /Switch/
SysVolSet   ARCHIVE      FDT       HP-IB     HP-IB
+---1----+ +---2----+ +---3----+ +---4---+ +---5---+
|2/4.3.0 | |6/4.4.0 | |2/8.0.0 | |2/8.1.0| |6/4.3.1|
|HPC2252A| |7937FL  | |HPC2252A| |C1511A | | 7980A |
|MEMBER1 | |MEMBER8 | |MEMBER80| |LDEV 36| |LDEV  7|
|LDEV   1| |LDEV   8| |LDEV  80| |tape   | |tape   |
|disc,   | |disc,   | |disc,   | +---+---+ +---+---+
|disc1   | |disc8   | |disc80, |     |         |
+--------+ +--------+ +array---+ +---+---+ +---+---+
|2/4.3.0 | |6/4.4.1 | |2/8.0.1 | |2/8.1.1| |6/4.3.5|
|HPC2252A| |7937FL  | |HPC2252A| |C1511A | |7978A  |
|MEMBER2 | |MEMBER9 | |MEMBER81| |LDEV 37| |LDEV 33|
|LDEV   2| |LDEV   9| |LDEV  81| |tape   | |tape   |
|disc,   | |disc,   | |disc,   | +-------+ +---+---+
|disc2,  | |disc9   | |disc81, |               |
+spool---+ +--------+ +array---+ +---6---+ +---+---+
           |6/4.4.2 | |2/8.0.2 | |2/4.0.7| |6/4.3.6|
           |7937FL  | |HCC2252A| | 2680A | |C1511A |
           |MEMBER11| |MEMBER82| |LDEV 16| |LDEV34 |
           |LDEV  11| |LDEV  82| |laser  | |tape   |
           |disc,   | |disc,   | +-------+ +---+---+
           |disc11  | |disc82, |               |
           +--------+ +array---+           +---+---+
           |6/4.4.3 |   UTIL               |6/4.3.4|
           |7937FL  | +--------+           |C5111A |
           |MEMBER12| |2/8.0.3 |           |LDEV 35|
           |LDEV  12| |HPC2252A|           |tape   |
           |disc,   | |MEMBER70|           +-------+
           |disc12  | |LDEV  70|
           +--------+ |disc,   |
           |6/4.4.4 | |disc70, |
           |7937FL  | +array---+
           |MEMBER14|
           |LDEV  14|
           |disc,   |
           |disc14  |
           +--------+
           |6/4.4.5 |
           |7937FL  |
           |MEMBER15|
           |LDEV  15|
           |disc,   |
           |disc15  |
           +--------+
```

How Messy Is My Database

Paper #5017

By David J. Greer

Robelle Consulting Ltd.
Unit 201, 15399-102A Ave.
Surrey, B.C. Canada V3R 7K1
Phone: (604) 582-1700
Fax: (604) 582-1799

How messy are things inside your databases? To answer this question there are two programs, DBLOADNG (available from the contributed library) and Robelle's HowMessy, which print a report on the internal efficiency of an IMAGE/SQL database. When DBLOADNG and HowMessy were first written, computers ran at a fraction of their current speed, MPE/iX didn't exist, and a large database had 100,000 records.

A messy database consumes more disc input/output (I/O). Despite the increase in CPU speed and disc space, these extra disc I/Os have a significant impact on system performance. This paper explains the information from the DBLOADNG/HowMessy report and suggests solutions for common database performance problems. To fully understand the report, you will require some background on the internal data structures of IMAGE.

Reference Material

One of the best references on IMAGE is **The IMAGE/3000 Handbook** written by Robert Green, Alfredo Rego, Fred White, David Greer, and Dennis Heider. You can order copies from Wordware, P.O. Box 14300-T, Seattle, WA 98114, fax (206) 222-5232.

IMAGE Background

All IMAGE datasets are separated into blocks. The size of these blocks is controlled by the $CONTROL BLOCKMAX parameter of DBSCHEMA. Most IMAGE databases have blocks that are 512 words long, perhaps 1024 words on MPE/iX. Each block contains one or more records; the actual number of records per block is known as the blocking factor.

Why are blocks important? All input and output to an IMAGE dataset is done by individual blocks and not by records. Because I/Os are the limiting resource on most HP 3000 systems, we wish to minimize the number of I/Os required to access our databases.

When we refer to a "messy" database, we mean one that takes more I/Os than the optimal case. The DBLOADNG/HowMessy report quantifies many of the I/O operations that take place in your databases.

Below are some tips on how to analyze the reports generated by DBLOADNG and HowMessy. The report is usually printed in 132 columns, but we will show it "folded".

Master Datasets (Ato or Man)

Your performance goal for master datasets should be one disc read for each hashed DBGET (i.e., mode-7) and three disc I/Os for each DBPUT (one read, one write, and one update file-label). IMAGE uses a hashing algorithm to translate your key values (e.g., CUSTOMER-NUMBER) to a number between 1 and the capacity of your dataset. Unfortunately, no hashing algorithm is perfect and IMAGE will assign the same record number to some of your key values.

In IMAGE terminology, we call one of these records a secondary. Most master datasets have some secondaries. We should only be concerned when there are many secondaries or when the secondaries are in different blocks (causing more I/O).

Example Reports - Master Datasets

```
                                              Secon-  Max
                 Type               Load  daries Blks  Blk
   Data Set           Capacity Entries Factor (Highwater) Fact
MCUST            Ato     21881    17295  79.0%  31.1%     6  28  ...
```

```
                  Max    Ave    Std   Expd   Avg   Ineff Elong-
...  Search Item  Chain  Chain   Dev  Blocks Blocks Ptrs  ation
   CUSTOMER         6    1.45  0.71   1.00   1.08   6.1%  1.08
```

The **secondaries** column is the percentage of all entries that are secondaries. **Maximum blocks** is the worst-case number of blocks that DBPUT will have to read to find the next free entry.

Secondaries for the same IMAGE record number are linked together into a chain. The **maximum chain** is the worst case for the dataset. If a secondary entry is in a different block from the primary entry, DBGET will have to do an extra I/O to retrieve your record.

Expected blocks is the best case (which is usually 1.00 for master datasets) while the **average blocks** are the number of blocks needed to hold the **average chain**. The **inefficient pointers** column is the percentage of secondary entries that span a block boundary (records in the same block are not a problem).

Diagnosis for this example? The MCUST dataset is healthy. There are not too many secondaries and most of them are in the same block.

```
                                              Secon-  Max
                   Type                 Load  daries Blks  Blk
      Data Set         Capacity Entries Factor (Highwater) Fact
    MSTATE         Man    18401   14722  80.0%  31.4%   15   11  ...

                     Max    Ave    Std   Expd    Avg  Ineff Elong-
    ...  Search Item Chain  Chain   Dev  Blocks Blocks Ptrs  ation
      ACCOUNT           7   1.46   0.72   1.00   1.31  23.5%  1.31
```

Our second example has about the same number of secondaries as the first, but there are two significant differences. The maximum blocks is 15 instead of 6, and the inefficient pointers is 23.5% instead of 6.1%. This dataset is less efficient than the previous one, even though it has effectively the same percentage of secondaries, maximum and average chain length and **standard deviation**. The reason is easy to see: MSTATE has a **blocking factor** of only 11 records per block, as opposed to the 28 per block in MCUST.

Master Dataset Solutions

- If secondaries are over 30% and inefficient pointers are over 50%, the dataset is either too full or not hashing properly. Increase **capacity** to a higher odd number, or change the data-type and format of the search field.
- Increasing the blocking factor should reduce inefficient pointers.
- Look for clustering problems: **load factor** less than 80%, secondaries less than 5%, and maximum blocks greater than 100. Clustering usually results from a sequential binary key; change it to type X, U, or Z.
- Changing a key's type, value, or format is almost impossible after a large database is in production. Build your test databases early and fill them with an appropriate set of values (or build your entire application database early if you have the resources). Use DBLOADNG on your test database. Early database changes are always the easiest to implement.

Detail Datasets (Det)

DBLOADNG/HowMessy reports chain information for each path in a detail dataset. Before accessing chains of records in a detail dataset, you must do a DBFIND for the path you are interested in. The DBFIND has the same overhead as a keyed DBGET (i.e., mode-7) so that poor master dataset performance will affect your detail dataset lookups.

Example Reports - Detail Datasets

```
                                           Secon- Max
             Type                 Load     daries Blks  Blk
  Data Set       Capacity Entries Factor  (Highwater) Fact
 DNAME       Det    21876   17295  79.1% (    18336)  4  ...

                 Max    Ave    Std   Expd   Avg   Ineff Elong-
 ... Search Item Chain  Chain  Dev  Blocks Blocks Ptrs  ation
    !CUSTOMER       1   1.00  0     1.00   1.00    0 %  1.00
   S ACCOUNT      245   1.17  3.14  1.03   1.17 99.6%  1.14
```

The secondaries and maximum blocks columns do not apply to detail datasets. In this example, the CUSTOMER path has one and only one record for each customer. We know this because the maximum chain length and the average chain length are one. Many IMAGE detail datasets have this same one-to-one relationship. If you find a maximum chain length of two, you know that there is at least one invalid record in your dataset. The inefficient pointers are 0% because there are no chain pointers in a chain length of one.

The ACCOUNT path has at least one account number for which there are 245 records, but on average there are 1.17 records for each account number. **Elongation** tells how inefficiently chains are packed, relative to their optimum packing. Most ACCOUNT chains are inefficient, with 99.6% of the existing pointers pointing to different blocks, but the elongation is small because the average chain length is small.

One problem with this dataset is that CUSTOMER was chosen as the primary (!) key. Even if the CUSTOMER path is accessed more often than the ACCOUNT path, it should not be the primary path because it has no chains. Instead, ACCOUNT should be made the primary path.

```
                                          Secon-  Max
                Type                Load  daries Blks   Blk
  Data Set         Capacity Entries Factor (Highwater)  Fact
DRECV           Det    208196  118609 57.0% (   155612)  23  ...
```

```
                    Max    Ave    Std   Expd   Avg  Ineff Elong-
...  Search Item  Chain  Chain   Dev  Blocks Blocks Ptrs  ation
 S!ACCOUNT         1604   8.06 35.75   1.36  11.32  72.5%  8.34
```

This dataset has only one path, ACCOUNT, which has an average of 8.06 records for each ACCOUNT, but the largest account has 1604 transactions. The standard deviation is 35.75 indicating that there are many accounts with three or four times the average.

This path is the first one with a large elongation value (8.34). This high value indicates that the DRECV dataset could be made much more efficient by repacking the entries that have the same key value. As the dataset stands, the average chain actually traverses 11.32 blocks, and 72.5% of the pointers within these blocks are inefficient.

This situation arises naturally over time. The dataset entries are loaded in date order, not in customer order, since new invoices and payments are loaded daily. This dataset is eight times less efficient than it need be, but the improvement gain by repacking would only be temporary. This dataset will need periodic repacking to maintain efficiency.

Detail Dataset Solutions

- Ignore load factor, unless dataset overflow is likely.
- If a detail dataset has more than one path, check that the primary path (!) has a large average chain length and is often accessed.
- Elongation of eight on a primary path means that disc I/O will be reduced by a factor of eight if you reload the dataset. You can reload a dataset using DBUNLOAD and DBLOAD, SUPRTOOL from Robelle, DBMGR from D.I.S.C., DETPACK from Adager, DBGENERAL from Bradmark Systems, or Flexibase from Proactive Systems. DBMGR, DETPACK, DBGENERAL, and Flexibase are the fastest.
- Look for average chain equal to 1.00, standard deviation about 0.01, and maximum chain of 2 or 3: this is usually a dataset that should have exactly one entry per key value but actually has some invalid duplicate key values.

- Look for paths with long chains (average chain plus standard deviation > 100), especially if the path is sorted (S).

- As with master datasets, build and test your database design early.

Estimating Response Time

This table shows the minimum number of disc I/Os required for each IMAGE intrinsic. An example follows of how to estimate response time, by computing the total disc I/Os for an example application.

Procedure	Disc I/O
DBFIND	1
DBGET	1
DBBEGIN	1
DBEND	1
DBUPDATE	1
DBPUT	3 + (4 x #paths, if detail)
DBDELETE	2 + (3 x #paths, if detail)
DBGET-2(Master)	Capacity/Blocking Factor
DBGET-2(Detail)	#Entries/Blocking Factor

Suppose that you want to add 100,000 records to a detail dataset with two paths. We assume that on a Classic HP 3000 you can achieve 25 disc I/Os per second and on MPE/iX you can achieve 50 disc I/Os per second:

3 + (4 x 2) = 11 disc I/Os per DBPUT

100,000 x 11 = 1,100,000 disc I/Os

Classic:
1,100,000 disc I/Os / 25 = 44,000 seconds

44,000 seconds = 12.2 hours

MPE/iX:
1,100,000 disc I/Os / 50 = 22,000 seconds

22,000 seconds = 6.1 hours

Automating HowMessy Analysis

For years, I was irritated that I had to read the HowMessy report once a month (or once a week or once a day, depending on the database). A lot of the analysis was routine and could easily have been done by a computer. To handle this problem, recent versions of HowMessy now create a self-describing file with all the data on the report. A self-describing file is an ordinary MPE file with information about the fields in the file.

With this file, you can do the following:

- Write a program (COBOL, PowerHouse, etc.) to read the file and analyze the report.
- Have products like Suprtool or AskPlus read the self-describing file to produce reports or control job streams.
- Write job streams that automatically fix the database, based on results found in the self-describing file (e.g., repacking detail datasets).
- Keep historical data so that trend analysis is possible.
- Have job streams monitor databases and use electronic mail to warn system administrators of potential problems (e.g., a dataset that is about to fill up).

Summary

All IMAGE databases become messy (unless they never change!) HowMessy and DBLOADNG provide a report that lets you analyze the efficiency of your databases. Understanding the report is easy, once you understand some of the internals of IMAGE. Then, constant monitoring of how messy your databases really are, will result in top performance from your IMAGE applications.

Copyright Robelle Consulting Ltd. 1993

Permission is granted to reprint this document (but not for profit), provided that copyright notice is given.

Paper #5018

CLIENT-SERVER WITH MPE - THE SEQUEL

Joseph C. Geiser
Insurance Data Processing, Inc.
One Washington Square
Wyncote, PA 19095

(215) 885-2150

INTRODUCTION:

Last year, this paper centered around how an MPE shop could take advantage of Client-Server technologies available then. Many things have happened over the past year to make this job even easier for anybody wanting to take advantage of Client-Server technologies. This paper will center on various methods on either writing new applications, or migrating existing legacy applications over to the Client-Server platform. Much of this information can even be used to evaluate existing applications which you may be considering bringing in.

Unlike last year, however, we won't center on a particular application or case study. We will look at alternatives and strategies, as well as coding examples, on creating efficient Client-Server applications.

EXISTING APPLICATIONS

Looking at existing applications on HP3000's, they haven't changed much over the last year. Most of the activity in Client-Server applications is taking place at the file server/client PC level. HP3000's are still running the same host-based applications that have been running for years, with a few exceptions.

Are you using your HP3000 to its fullest potential?

- Do you have intensive, online transaction-based systems?
- Has your response time degraded, with the addition of new users or function?
- Have you considered upgrading your CPU, because it cannot keep pace?
- Have you been replacing your terminals with PCs?
- Are you looking to add function, like document imaging, document archival, OCR, etc.?
- Do you want to implement Graphic User Interfaces within your systems?

If the answer to one or more of these questions is yes - then the time is now, to start looking at your current and planned applications for possible migration to a Client-Server platform. Why? Let's look at each item, and why a client-server solution would be appropriate for each:

1. Do you have intensive, online transaction-based systems?

More than likely, your systems are online and transaction-based, after all, the HP3000 and TurboIMAGE are great at this type of application. The database is fast, the system responds better to online use than with concurrent batch use. But where is the work being done? On the HP3000. As users are added, the system slows down due to the increased workload. It may not be noticeable at first, but it will be - and sooner than you think!

The majority of the workload could be placed on a PC, and relieving the HP3000 to perform the managing of data. You will get better performance, with more users connected, with the same computer.

2. Has your response time degraded, with the addition of new users or function?

You bought this great HP3000, it performs better than your wildest expectations. You add a new application to it, then another, then another. More users are added and it starts to slow down. Remember, the more you add to the same system, the time necessary to process the increased workload will increase. Simple common sense, right? We all think so, but those who run our companies don't necessarily see it that way. Dollars for upgrades are not as plentiful as they were in days gone by.

Another factor to consider is Batch Processing. Although it usually runs in a lower queue, high-volume summarization of data can often wreak havoc on online users. Summarization of data for daily and monthly processing can be handled differently, such as summarization "on the fly," as each transaction is processed. This type of summarization can provide more benefits than imagined, such as the implementation of Executive Information Systems (EIS) and online reports, as opposed to the daily and monthly reams of paper, which become obsolete in a matter of hours or days. This is accomplished by sharing the load between the PC and the HP3000. Who knows, you might even save a tree or two in the process as well.

3. Have you considered upgrading your CPU because it cannot keep pace?

So, now you inform the folks with the checkbook that you need an upgrade. This could cost anywhere from $10,000 on up, depending on what your upgrading to. Lord help you if you have a Classic (Series 70 and below). Can you justify this cost?

You could upgrade the applications and keep the same machine, at a lower cost. The same computer could be executing just one application (data serving), with the bulk of the application on the PC. It can be done, and with little retraining of existing staff, and with tools that are quite inexpensive.

4. Have you been replacing terminals with PCs?

PCs have proliferated to the point where they are now quite inexpensive. It is getting unusual to see terminals on desks anymore, But what are these PCs running? Reflections, Sessions, AdvanceLink, Minisoft's MS-92, whatever - it's a termulator, and termulators just make a $1,300 PC act like a $600 terminal.

That PC probably has at least a 1MB of RAM, possibly more. It has a powerful processor. It may have a network adapter which makes it run at 10 megabits. Why just use a PC as a glorified terminal, when you can perform the bulk of your processing locally, and reserve that speed for data access from the HP3000, and use the HP more efficiently.

5. Are you are looking to add function, like document imaging, document archival, OCR, etc.?

There are some things that do not work well with the HP3000. File servers dedicated for the purpose of handling these tasks are far superior solutions.

Document imaging is one example. Even the solution used by HP when implementing this function with existing and new customers uses either an HP9000 or a high-end Vectra server - both UNIX-based, and NOT HP software!

There are other systems which work equally well in a Local Area Network environment, which are Novell, DOS and Windows based. If your application is HP3000-based, you will find it difficult to integrate this function.

6. Do you want to integrate Graphic User Interfaces within your systems?

Many companies are finding that Graphic User Interfaces (GUIs) provide not only a great way to manage applications, but with the advent of data exchange and object orientation and linkage - applications such as word processors and spreadsheets can be literally linked in - integrated - into new and migrated legacy applications seamlessly. It's more than just a glorified menu system, folks - there are real advantages here!

TO MIGRATE, OR NOT TO MIGRATE? THAT IS THE QUESTION!

Well, do you migrate, or keep the same old stuff? The answer to this lies in your particular application. The first step in ascertaining whether a re-engineering effort is worth it is to study all aspects of all applications used in your company. Here are some things to look for:

- Online applications are better candidates than batch applications
- Applications which do not perform a great deal of updating in one logical transaction (five to six physical writes per one logical transaction for TurboIMAGE is a guideline, although not a rule)
- Applications which can be easily ported to an "object orientation". An example of this would be the ability to break the application into smaller objects (copylibs, database I-O, screen handling)
- Applications which fit one or more of the above criteria, and are considered "mature" and are prime rewrite candidates.

Why is this criteria important?

Client-server applications run on the PC, not the host. The host merely supplies data, and herein lies the problem. The host must filter that data and send it to a PC, which then in turn, processes it, and either requests the next record from the host, or updates the existing record and requests another record.

Batch processes are more efficient as host-based processes, because the data stays in the system, usually requires large volumes of data, as opposed to a limited set of records, and therefore, can process it much quicker than a PC.

On the other hand, online applications usually deal with a limited set of data, perform direct keyed or hash reads against keyed files (table lookups) and write logical transactions consisting of an average of 5 to 6 physical file writes. For calculation intensive, or online work, it's more efficient for a PC to perform this work, relieving the host from these duties.

The best place to start is a study. The group to study the issue should not only contain management, but programmers and analysts as well, as they know the internals of the applications better than anyone else. If your staff is limited, then outside consultants can be used to give an objective, third-party assessment of your applications, and the chances of success in migrating applications to a client-server environment.

TOOLS OF THE TRADE- WHAT'S AVAILABLE? (...and are they cheap?)

Tools are in abundant supply today. It seems that all the major players are providing client-server tools to speed delivery of new systems, and the cost is coming down.

The keys to developing these systems are:

- **Host Filters** - Host-based procedures in SLs/XLs which are integrated into the new system to filter large amounts of data, and minimizes network traffic as well as unnecessary PC processing.
- **Reusable Code** - Building code objects, which can be reused in other programs.
- **Speed** - Average application turnaround should not exceed eight months.
- **Ease of Training** - Tools must be easy to learn and use.
- **Maintainability** - Results must be easy to maintain.
- **Portability** - Applications should be object oriented, so that migration to other platforms does not mean a COMPLETE rewrite, just certain code objects - the application should remain intact.

Most client-server applications being developed today are based on Microsoft Windows. To many, the thought of writing code for Windows is scary, unless they've trained for it. This misconception is brought about by the fact that people are under the impression that it's C/C++, or nothing. The fact of the matter is, that most "front-end" or online portions of Windows-based programs are now written with either Visual Basic or Visual C/C++ (Visual Basic is the most common) or using Windows-based 4GLs such as Actor, Powerbuilder or ObjectVision.

Visual Basic, is by far, the easiest to learn, and is very powerful - this is not the BASIC you grew up with in High School and College - this is a professional version which can do anything that can be done with C/C++. Small applications can be developed in hours. Large applications, which in a traditional language such as COBOL would take over a year to develop, can be done in six months or less with Visual Basic.

The client-server aspect of any application uses "middleware" - a layer of code which resides between your application and the host server. There are several middleware applications available for the HP3000, some of which are:

- Process-to-Process Link (PPL) for Windows (and DOS) from WRQ
- Omnidex Client-Server (based on PPL, but with Omnidex Links)
- EDA/SQL from Information Builders
- Third-Party Database Mangement Systems such as Oracle and Sybase
- ...and yes, HP Cooperative Services is back on the CPL, and we'll see for how long, and what improvements are planned from the SWT division.

Which middleware product you would want to use depends on your plans for the future, your desire for "open systems" and your budget (or lack thereof...).

PPL has two toolkits. Standard PPL requires that a host server be custom written by the user, which would perform all actions required by the client PC. This portion is portable between the HP3000, the DEC/VAX and Unix-based computers. The PPL HP Toolkit is for standard HP3000 access and TurboIMAGE. It provides for all of the TurboIMAGE intrinsics and many of the standard intrinsics. It also provides for Remote Procedure Calls (RPCs) - procedures in SLs/XLs.

Omnidex Client-Server was based on PPL, however, extensions were added for current (and new) users of Omnidex. If you currently use Omnidex Indexing, then you can get the advantages of PPL, with the Omnidex extensions in this version.

EDA/SQL is the "open" solution. Information Builders has put together a product, which will allow an application to use a standard set of SQL calls, and access over 30 different database management systems, including Allbase and TurboIMAGE, as well as those on IBM platforms (DB2, OS/400) and PC/Server types like X-Base - and many others. If you have a multi-vendor environment, this is worth checking out.

The areas of concern for EDA/SQL are price and training. There is a set price which covers the core EDA/SQL software and one DBMS. Additional charges are assessed for additional DBMS support. Training is an issue, as SQL is used exclusively, even with TurboIMAGE databases.

Oracle, naturally supports Oracle. If you're an Oracle user, then you probably already know it's capabilities. Using Oracle, your application will be supported unchanged on over 70 different hardware platforms - from PCs to Mainframes. Oracle's TCP/IP-Connect will permit true client-server connectivity with any supported platform including the HP3000.

The main concerns here are again, price and training. Oracle runtime on a PC is quite inexpensive, but the HP3000 implementation can be a six-figure proposition. As for training, SQL is used exclusively, and the intricies of Oracle will need to be learned as well.

SO, HOW IS THIS ALL PUT TOGETHER - HOW DO I DO IT?

The common implementation of all of these products, with the exception of Oracle (as of this writing), is that they are implemented using Dynamic Link Libraries (DLLs) on the PC. DLLs are the PC equivilent of SLs or XLs, however, they are used during execution, not load or link time. The DLLs are also sharable by multiple users. To access a DLL module, you merely declare its use in the program, and make a call to it (see figure 1).

```
Declare Sub ACTIVATE Lib "PPLHTK.DLL" (ByVal pin%, ByVal susp%)
Declare Function CALENDAR Lib "PPLHTK.DLL" () As Integer
Declare Sub CALLPROC Lib "PPLHTK.DLL" (ByVal plabel%, ByVal inbuf$, ByVal inlen%, ByVal outbuf$, outlen%)
Declare Function CLOCK Lib "PPLHTK.DLL" () As Long
Declare Sub COMMAND_HP Lib "PPLHTK.DLL" Alias "COMMAND" (ByVal coimage$, xerror%, param%)
Declare Sub CREATE Lib "PPLHTK.DLL" (ByVal progname$, ByVal entryname$, pin%, ByVal param%, ByVal flags%)
Declare Sub CTRANSLATE Lib "PPLHTK.DLL" (ByVal code%, ByVal instring$, ByVal outstring$, ByVal slen%)
Declare Sub DATELINE Lib "PPLHTK.DLL" (ByVal datebuf$)
Declare Sub DBBEGIN Lib "PPLHTK.DLL" (ByVal xbase$, ByVal text$, ByVal mode%, dbstatus%, ByVal tlen%)
Declare Sub DBCLOSE Lib "PPLHTK.DLL" (ByVal xbase$, ByVal dset$, ByVal mode%, dbstatus%)
Declare Sub DBCONTROL Lib "PPLHTK.DLL" (ByVal xbase$, ByVal qualifier$, ByVal  mode%, dbstatus%)
```

(Figure 1 - Declarations of DLL Procedures - PPL with Visual Basic)

You will notice, that the procedures defined in the example above should look very familiar. Once defined, these procedures can be called from any point in an application. An example is shown in figure 2, below:

```
Function Find_dLines (KeyItem As String) As Integer
   dbSet = "D-LINES;"
   dbList = KeyItem

   DBFIND clmDBbase, dbSet, 1, dbStatus(0), dbList, wsClaimantNum, 18

   If dbStatus(0) <> HTK_SUCCESS Then
      Find_dLines = dbStatus(0)
      Exit Function
   End If
```

(Figure 2 - Implementation of DBFIND using PPL in Visual Basic)

In this example, the DBFIND is written just as it would be written on the HP3000, with the exception that it contains one additional parameter - the length of the search item. It is also implemented as a Reusable Function, and therefore, is called from within the application in multiple areas. An example of how it is called is shown in figure 3, below:

```
xStatus = Find_dLines(KeyItem$)
If xStatus = 17 Then
   MsgBox "No Entry Found", 32
   Exit Sub
End If

xStatus = 0            'reset the status code

Do
   KeyItem$ = "CLAIMANT-NUM;"
   xStatus = Get_dLines()

   If xStatus <> HTK_SUCCESS Then Exit Do

   Entry$ = Str(x) + TB
```

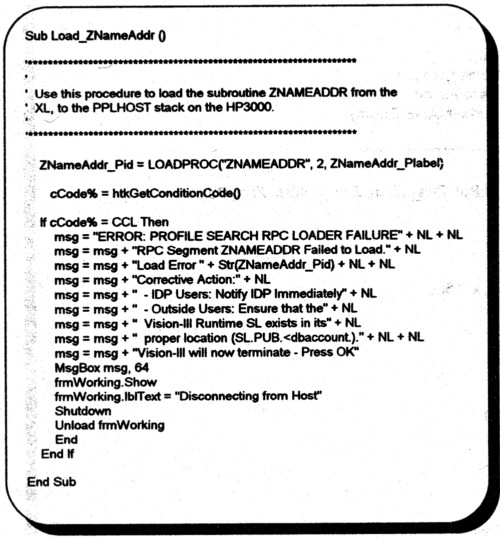
(Figure 3 - Calling Reusable Functions)

In this code fragment, the variable "xStatus" is used for the return of the TurboIMAGE Condition Word. The function name is descriptive, and the explicit TurboIMAGE intrinsic is removed from the main body of the application. The Get_dLines function shown further down references a DBGET in the same manner. (Please note that this code fragment is not complete, but shows enough for explanation).

Remote Procedure Calls to the Host

In addition to moving code down to the PC - there are areas in many applications where industry-specific formulas, or code which have been used for years would, if rewritten, probably not work the same way as it did on the host-based system. A perfect example in our system is our implementation of Soundex Name Search.

To solve this problem, we kept the same code we've been using for years - it works, and can continue to be used. We did modify it to work in a client-server environment. The modifications took about 2 hours to perform. Most of the tools shown here have the capability of executing Remote Procedures. The figure shown below illustrates loading a Remote Procedure using Visual Basic and PPL. The procedure is a segment in an SL called ZNAMEADDR. It is loaded on the stack of the PPL Host Server process, called PPLHOST.PUB.SYS, and remains there until unloaded.



```
Sub Load_ZNameAddr ()

'*************************************************************
'
' Use this procedure to load the subroutine ZNAMEADDR from the
' XL, to the PPLHOST stack on the HP3000.
'
'*************************************************************

 ZNameAddr_Pid = LOADPROC("ZNAMEADDR", 2, ZNameAddr_Plabel)

  cCode% = htkGetConditionCode()

 If cCode% = CCL Then
    msg = "ERROR: PROFILE SEARCH RPC LOADER FAILURE" + NL + NL
    msg = msg + "RPC Segment ZNAMEADDR Failed to Load." + NL
    msg = msg + "Load Error " + Str(ZNameAddr_Pid) + NL + NL
    msg = msg + "Corrective Action:" + NL
    msg = msg + "  - IDP Users: Notify IDP Immediately" + NL
    msg = msg + "  - Outside Users: Ensure that the" + NL
    msg = msg + "  Vision-III Runtime SL exists in its" + NL
    msg = msg + "  proper location (SL.PUB.<dbaccount.)." + NL + NL
    msg = msg + "Vision-III will now terminate - Press OK"
    MsgBox msg, 64
    frmWorking.Show
    frmWorking.lblText = "Disconnecting from Host"
    Shutdown
    Unload frmWorking
    End
 End If

End Sub
```

Figure 4 (next page) illustrates the call to the Remote procedure, and the Unload as well.

Another area where Remote Procedures are beneficial is in the use of Data Filtering. An example of this would be a large chained read. If a dataset access will cause a large chain read, it may be beneficial to implement a filtering mechanism on the host.

The reason for using a remote procedure to filter data coming from the host is simple. For large data accesses, that data must make its way to the PC, which in turn either uses the record, or rejects the record and requests another. The Host-Based Remote Procedure Filter would be a more efficient approach, as it can process these records faster, and then only pass to the PC, those records which qualify for processing, and relieve the PC of processing records which would be rejected.

The first way would be to write a custom remote procedure, which would return only records which fit a certain criteria. This is desirable when the data requested is specialized, and/or the access is cumbersome.

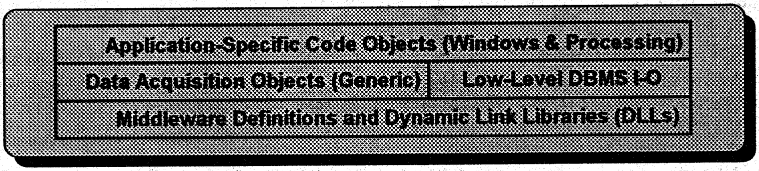
In addition, PPL has implemented a DB call for TurboIMAGE called DBSEARCH. It works in conjunction with DBFIND/DBGET, and allows for equalities as well as greater/less than comparisons, and then stores the record numbers on the host for later DIRECT retrieval. Omnidex can also help in this area with their indexing schemes implemented into your databases.

```
Function Call_ZnameAddr () As Integer
'********************************************************************
'
' Use this procedure to call the subroutine ZNAMEADDR from the
' XL, and return 6 name/addresses at a time from the HP3000.
'
' Return Values are as follows:
'
'    0 = Successful, More Names Available
'    1 = Successful, No Names on List, Name not Found
'    2 = Successful, End of List, Names were Found
'    3 = Unsuccessful - FUTURE USE (TurboIMAGE Errors)
'
'********************************************************************

  CALLPROC ZNameAddr_Plabel, Zna_InBuf, Zna_InLen, Zna_OutBuf, Zna_OutLen

  z = 1

  For x = 1 To 791 Step 158
    Zna_OutEntry(z) = Mid$(Zna_OutBuf, x, 158)
    z = z + 1
  Next x

  Call_ZnameAddr = 0

  If Left$(Zna_OutEntry(1), 2) = "Z1" Then
    Call_ZnameAddr = 1
    Exit Function
  End If

  For x = 1 To 6
    If Left$(Zna_OutEntry(x), 2) = "Z2" Then
      Call_ZnameAddr = 2
      Exit For
    End If
  Next x
End Function

'*******************************************************

Sub Unload_ZNameAddr ()
  UNLOADPROC (ZNameAddr_Plabel)
End Sub
```

(Figure 4 - Calling a Remote Procedure - Visual Basic and PPL)

Object Management

Using Object Orientation, applications are created faster, and code becomes reusable. This is the main feature of using Object Oriented Programming. The particular model we use is shown in figure 5, below:


Application-Specific Code Objects (Windows & Processing)
Data Acquisition Objects (Generic)
Low-Level DBMS I-O
Middleware Definitions and Dynamic Link Libraries (DLLs)

(Figure 5 - Client-Server Object Model)

The "Application" layer of this model supports all code objects which deal directly with window handling and application-level processing. Examples of this type of processing would be:

- Showing, Hiding and Editing of Windows and Fields
- Internal Calculations
- Data Buffer Manipulation
- Conditional Data Editing and Manipulation
- Data Collection
- Error Handling

The "Data Acquisition" layer of the model encompasses all code objects, which are generic in nature. An object in this layer may perform a data retrieval for a "logical data unit", such as policy data for an insurance application, or a complete bill of materials for a particular part in a manufacturing application. Although the data itself may reside in multiple datasets or databases, this code object would perform the I-O operations necessary to retrieve the entire "unit", and pass it back to the application.

In the example for acquiring policy data, an object (or function) might perform the following functions:

- Edit as to whether a policy actually exists on a policy master file
- If the policy exists, retrieve the policy records for a particular policy period
- Retrieve all premium transactions for the policy period
- Retrieve all coverage limits and endorsement information related to the policy
- Return to the application

In this manner, the application needs only call the data acquisition function, and the necessary data is returned, or an error status code to indicate errors, such as a non-existent policy.

This layer is optional, however, if the data to be accessed can be found on one dataset. The example shown above is such an example.

The "Low-Level Database I-O" Objects perform the actual data access, and interface directly with the middleware. These objects should be very limited in scope, in that they should perform one operation, against one dataset (for TurboIMAGE) or one database (for other relational databases). These functions should include:

- All code necessary to successfully interface with the middleware objects
- All low-level error handling routines, but should pass status information back to the calling object, not directly handle error messages, etc.

An example of this type of object is found in Figure 2, previously shown. This code object, "Find_dLines", performs a single DBFIND (which is defined in the middleware interface), but only returns an integer, containing the condition code returned from TurboIMAGE. The code object found in Figure 3, where the line:

```
xStatus=Find_dLines(KeyItem$)
```

is shown, the next IF block actually takes the status returned to "xStatus", interprets it, displays the error message, and takes the necessary action.

The Middleware definitions layer is the lowest layer in the application. It contains all of the definitions for all functions to be used within the other layers - specifically the Low-Level Database I-O layer. An example of this layer is shown in Figure 1, above.

So, why all of this object orientation? Could you not imbed the code directly into the application? Yes, you could, and the same results would be achieved, with the same speed as breaking it up into objects, however, the advantages of coding a separate layer far outweigh the disadvantages (for which I have found none so far):

- Breaking Low-Level code from the Application Code allows better portability, in that changing to another database, if not using EDA/SQL, or some other generic middleware, would require recoding of only the low-level I-O level - the application layer remains entirely, or mostly, intact. Coding low-level objects in-line with the application would mean changing the application - and all occurrances within the application where these calls are taking place.

- The maintenance of the application is much clearer. Debugging an object-oriented application is quicker, because the problem is usually localized in one code object, which is usually small and easy to work with.
- Because the same dataset is probably accessed in multiple areas within an application, the same low-level database object can be used in all areas. An enhancement or modification to this low-level object is applied to all programs referencing it, again, without modifying one line of code in the application.
- Other applications which need to reference these datasets can use this object as well. It comes down to whether you write this code multiple times - or just once, and reuse it as needed. Again, maintenance to the low-level object is reflected in all applications which reference it. No more changes "falling through the cracks".

In essence - object oriented programming in a client-server environment permits the creation of tighter code, localized function and an overall better-quality product.

Speed - Getting it done quickly

How many times have you received a request, analyzed the request, estimated the time it will take to execute the request, only to hear from the end-user that "it's not good enough - I need it sooner than that!" Every user always needs his or her project done last week.

This is another benefit of reusable code objects, and the use of programming tools which facilitate the creation of prototypes, which can be easily modified and changed into production systems. This is precisely why Microsoft created Visual Basic, and subsequently created Visual C/C++, and why other 4GL vendors such as Whitewater (Actor) and Borland (ObjectVision) have created their solutions as well.

Like it or not, the development cycle is shrinking - rapidly. The days of full analysis, detail design specifications, signoffs, phased implementation and full signoffs are over. Implementation of major systems which take a year or more are being cut down to the six to eight month timeframes, and are shrinking. Analysis and specification has been replaced with Prototyping. The programming involves modifying the prototype system. Prototyping is increasingly being done with tools shown above. Major systems are being implemented in months, not years.

How is this occurring? Use of Visual tools like those shown here are enabling systems engineers (programmers, analysts, etc) to produce quality applications with end-user input along the way. Reusable code developed in previous projects are being used again, thereby reducing the coding time, and results of changes are instantaneous. The end-user can actually see the result while changes are being made. All of this results in better user satisfaction, in less time, with less work on the part of the systems engineer.

Ease of Training - Getting your HP staff up to speed quickly

This is usually a big problem when changing platforms, methodologies or procedures. People are naturally reluctant to change, or explore tools for which they are not familiar or trained.

In addition to the tools, the mere aspect of Object Orientation, is tough for a person who has coded in the "top-down" method for so long to grasp. Our experience has shown that the transition does not need to be painstaking.

For example, if you are choosing to stay with TurboIMAGE, and utilize PPL or the Omnidex, your retraining time will be limited to the use of object oriented programming techniques. The TurboIMAGE database calls are the same, whether using COBOL or Visual Basic (with some very minor deviations for DBGET and DBFIND).

The tutorial which comes with both the Standard and Professional Editions of Visual Basic and Visual C/C++ are excellent primers and take a programmer, step-by-step through the process of developing object oriented code.

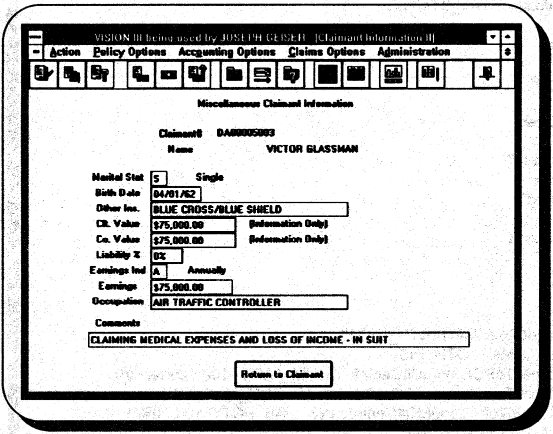
The "Windows" aspect of programming - the handling of calls to the Windows API, and the handling of the Windows Message Loop (the frightening part of Windows programming without adequate training), are a non-issue with the Visual products and other 4GLs. These issues are automatically handled within these products.

To illustrate the ease of showing a Window to the screen, the following code fragment gives an example using Visual Basic.

```
Sub btnClaimantMisc_Click ()

   frmMiscClaimant.Show

End Sub
```

The "Sub" is a subroutine, generated automatically by Visual Basic, in response to a user pressing the "Miscellaneous Claimant" button on the form. At this point, the form <u>**frmMiscClaimant**</u> is loaded and the "Show" method causes the window to be shown on the screen.

After the named form is referenced, Load and Activate procedures are executed for that form. This causes data to be initialized in the form. To illustrate how easy it is, Figure 7 below shows the Window itself, and Figure 8 (next page), is the actual code which loads the form:


VISION III being used by JOSEPH GEISER [Claimant Information II]
Action
Policy Options
Accounting Options
Claims Options
Administration
Return to Claimant

Miscellaneous Claimant Information

Claimant# DA00005003
Name VICTOR GLASSMAN

Marital Stat S Single
Birth Date 04/01/62
Other Ins. BLUE CROSS/BLUE SHIELD
Clt. Value $75,000.00 (Information Only)
Co. Value $75,000.00 (Information Only)
Liability % 0%
Earnings Ind A Annually
Earnings $75,000.00
Occupation AIR TRAFFIC CONTROLLER
Comments
CLAIMING MEDICAL EXPENSES AND LOSS OF INCOME - IN SUIT

Figure 7 - Miscellaneous Claimant Information Window

Maintaining Client-Server Systems

What you will find is that Client-Server systems, which employ an object oriented approach are easier to maintain, than their counterparts on mainframes. The reasons stem from the fact that small objects are used to build systems, instead of large, procedure-based programs.

What this means is that since each functional area is it's own code object, working with other code objects - errors are easier to find, and applying maintenance to these objects is much more manageable.

```
Sub Form_Load ()

   lblClaimantNumber = dClaimant.CLAIMANT_NUM
   lblClaimantName = RTrim(dClaimant.CLAIMANT_NAME)
   txtMaritalStatus = dCltGenInfo.MARITAL_STATUS

   Select Case txtMaritalStatus
      Case "S "
         lblMaritalStatus = "Single"
      Case "D "
         lblMaritalStatus = "Divorced"
      Case "W "
         lblMaritalStatus = "Widowed"
      Case "M "
         lblMaritalStatus = "Married"
      Case "X "
         lblMaritalStatus = "Separated"
      Case "  "
         lblMaritalStatus = "N/A"
   End Select

   txtBirthDate = YMDToDisplay(dCltGenInfo.BIRTH_DATE)
   txtClmntOthIns = dCltGenInfo.CLMNT_OTH_INS
   txtClaimAmt = Format$(HPZonedToPCLong(dCltGenInfo.CLAIM_AMT) / 100, "currency")
   txtValueAmt = Format$(HPZonedToPCLong(dCltGenInfo.VALUATION_AMT) / 100, "currency")
   txtInsLiabPct = Format$(HPZonedToPCLong(dCltGenInfo.INS_LIAB_PCT) / 100, "0%")
   txtEarningsFlag = dCltGenInfo.EARNINGS_IND

   Select Case txtEarningsFlag
      Case "H "
         lblEarningsFlag = "Hourly"
      Case "D "
         lblEarningsFlag = "Daily"
      Case "W "
         lblEarningsFlag = "Weekly"
      Case "M "
         lblEarningsFlag = "Monthly"
      Case "A "
         lblEarningsFlag = "Annually"
      Case "B "
         lblEarningsFlag = "BiWeekly"
      Case "S "
         lblEarningsFlag = "SemiMonthly"
      Case "  "
         lblEarningsFlag = "N/A"
   End Select
End Sub
```

(Figure 8 - Code used for load the Miscellaneous Claimant Form - Visual Basic)

For example, let's say that a user reports an error in the Claimant Window, shown above. The error was that the incorrect Claimant Name is appearing on the Window. Since each window comprises it's own code object, the programmer will know exactly which object to look into. The field in question is a display-only field, the Form_Load procedure would be the primary place to look.

Since each object is small, it should not take much time to correct the error at all, and if, in the event, the error extends to other objects in the same program, they can be easily accessed and corrected as well.

One of the nicer things about making corrections and enhancements to a code object, is that the same enhancement will apply to all other programs which use the same code object, once they are recompiled.

Portability of Code - Can I use it on other Platforms?

The answer to this question depends on your design. Using the object-oriented design techniques which were described in this paper (ie: separating the database I-O from the application), then all that would be necessary is to change the database I-O, in most cases. The application remains intact.

If you used a middleware product such as EDA/SQL, or Oracle, then you should be able to port your application to any other supported platform or database management system supported by these products, with little or no modification.

Design implications drive many factors of client-server products. An "open" system can be designed using almost any middleware, most of which depends on how the database I-O is implemented. If the design of the application objects is generic in nature, when accessing the databases, then at most, the database access objects would need to change, leaving the application intact. This would be true of PPL, the Omnidex variant of PPL, or even HP Cooperative Services.

Another area to consider in your design is your choice of development tools and user interface. As was stated earlier, the user interface of choice is Microsoft Windows. If you are going to support Windows, then the development tools should support Windows as well, including all of the criteria shown in this paper - object orientation, ease of training and use, maintainability, Remote Procedure access and speed.

Choosing to use solely C/C++ (not Visual C/C++), for example, in an environment where all of the programmers have been using COBOL, Fortran or even Transact, would yield a very large learning curve - even if they were to take a semester in C/C++ at a local college. Windows programming in C/C++ is very complex and causes even the seasoned programmer to think twice. In addition, development time will take many times longer than with tools available today. If you're writing your own Dynamic Link Libraries, then use C/C++ for that purpose, as the Visual tools cannot create these, but the Visual tools, and other 4GLs available today, can make the job of writing a Windows-based system easier than writing systems on the HP3000 in COBOL.

SUMMARY

HP and third parties have done quite a bit of work in the area of client-server programming and tools. They are more plentiful, more robust and less expensive than just one year ago.

Client-server applications take advantage of the power of the PC, which is ever decreasing in price, as well as the host. In other words, processing power is spread over multiple computers instead of just one, and each is doing what it does best. Efficient processing of data is the result.

The advantages of this technology are obvious. Host-based systems limit what can be done in implementing new technologies which make applications easier for the user to use, and easier for the programmer and analyst to design and build.

In addition, development cycles are shrinking, and for companies to remain competitive in today's rapidly changing environment, productivity and prototyping tools, are necessary to complete applications faster than ever before. Object oriented programming, as well as Visual and 4GL tools are being utilized to speed this development process.

Any company facing a reengineering effort of mature applications, contemplating a downsizing or rightsizing effort, or looking at equipment upgrades should seriously consider client-server applications. Serious time, money and effort can be saved by migrating systems to a client-server platform, as opposed to rewriting on a host, or needlessly upgrading host systems.

More and more, vendors are creating applications to take advantage of this technology. PeopleSoft, a payroll and human resource application is a very good example of how client-server systems are being implemented in critial areas on HP3000s, as well as other platforms). As time progresses, vendors will be embracing these techniques and more systems will emerge, operating under a client-server umbrella.

Paper # 5019

Getting Started With ALLBASE/SQL

John Schmid
3M Company
3M Center 224-3S-25
St. Paul, MN 55105-1000
612-733-9906

Summary

One of the biggest hurdles towards relational database is getting started. At 3M, we began our evaluation of ALLBASE/SQL in June 1989. At that time we had two sites that were using it. In 1991, we decided to introduce ALLBASE/SQL product support for a potential 100 sites. Today, usage of this product has finally taken off within 3M. We have an active internal Allbase User Group with over 90 members!

This paper presents a practical approach for the potential ALLBASE product champion. It tells how to introduce potential users to ALLBASE. This approach is generic enough for any relational database product.

The author is a database design analyst and provides a DBA perspective. Recognizing the diversity of the potential audience, two perspectives are presented: the single site offering and the multi-site offering.

Disclaimer

The products and vendors listed in this paper is intended solely for the convenience of the reader and is not to be interpreted as an endorsement by 3M or by the author of this paper.

Overview

The more experience I have with other relational databases, the more exited I get about Allbase/SQL. Allbase has a tremendous potential for any size organization and it's easy to use! However, persuading others to use it is not necessarily an easy task. In this paper, I will be presenting the steps I have taken at 3M that include the following:

1. Develop awareness
2. Get education
3. Focus on user needs
4. Develop a plan
5. Introduce support
6. Consider support
7. Consider migration

A lot of this paper contains some very obvious, common sense things to do. Hopefully, the reader will benefit by the way it is put together and use this as a checklist for your company.

Develop awareness

The first step to getting started is to develop your own awareness as to what is going on with Allbase. HP provides a manual called Up and Running with ALLBASE/SQL. In one hour, maybe 3 with interruptions, you can be using Allbase with HP's PartsDBE, a database available with the Allbase product on your system.

If you don't have a copy of Allbase yet, I would suggest reading available journals including the Interact, The HP Chronicle and the HP Professional. In addition, there are usually papers being presented at HP Regional User Groups or the Annual HP User Conference sponsored by Interex (check the above journals for the nearest conference to you).

If you have a local HP office, ask your sales representative about the available seminars for your area. Many of these seminars are low cost or free. In absence of a seminar, your HP support personnel may be of some help. Depending on the size of your site and how many people are involved, HP may send someone to your site to present information on Allbase.

Finally, join the Interex special interest group called SIGALLBASE/SQL. You may contact the Interex office or call Dana Brown at 415-382-2184. Be sure to ask them to send you a copy of the latest newsletter.

Get Education

HP offers some of the most comprehensive classes that can be offered on relational databases. If you plan to use or support Allbase, the class HP Allbase/SQL on HP-UX and MPE/iX is a necessity. This class might give you more than what you need to get started but the class notes provide good documentation for the future.

I strongly recommend that you use some form of data modeling for your database design. Up front data design will save you maintenance over the life of your application. According to recent studies, application maintenance represents up to 90% of the total lifetime cost of your application. HP's class, HP Allbase/SQL Database Design Theory, might give you a good start. 3M has a department dedicated to data modeling, so I haven't had need for HP's class.

Another alternative is to consider general classes on SQL and data modeling. Be sure to take advantage of any free seminars, classes and/or tutorials that are given at conferences.

Focus on User Needs

Without a focus on your user needs, the introduction of Allbase could be a fruitless activity. If users cannot relate Allbase to their business needs, they won't bother to give it a try. It would be worth while to spend time with potential users. Find out what their business goals are. Identify what their areas of dissatisfaction are with their current applications. Determine what their access requirements are. Then identify possible alternatives for meeting their requirements. Finally, prioritize these requirements with their help.

For me, this was no academic exercise. This gave me some clues as to how Allbase fit our user needs at 3M and how we could introduce Allbase to satisfy these needs.

Develop a Plan

The next logical step is to put together a plan of action. This must include where you are at currently and identify your direction and timing for introducing Allbase. The user goals and requirements identified above should be documented in a form that is understandable by your management. Next, define what services that you will provide including what you can or cannot support. Be sure to identify your goals and timing. Finally, your plan should include an estimate of the costs and benefits. Be sure to communicate how your plan fits your local budget and how it may effect the bottom line, if possible. The key point to make is what's in it for my users.

This can be a very difficult step, especially when you do not have much experience with relational databases. However, give it your best shot and make your plan a working document that will be revised as you move forward.

Introduce Support

The most critical step is obtaining management buy-in. If you have communicated the results of the above steps to your management all along, you already know whether or not you will be supporting Allbase. In most cases, your management has already decided and has assigned you the mission to introduce Allbase into your work environment.

Next, you need to set up a support team. This will consists of application leads, managers and technical support people. Be sure to include anyone who must contribute to the support of Allbase in your company.

A formal announcement should be made to present and future users of Allbase. This can be done in the form in a meeting, letter, report or internal newsletter. I highly recommend a meeting arrangement where you can interact with your audience and answer questions.

Finally, I highly recommend using Allbase in a small pilot project. This should be done as soon as possible after taking the Allbase class above. The experience will be extremely valuable for having something concrete to present to potential users. Consider using this project as a demo. A success story says a lot about Allbase as a viable product and will perk potential interest.

Consider Support -- Single Site

This section applies to both the single HP site and the multiple HP site company. I mention the single site first as most of the multiple site issues would not be applicable here. I have listed some questions for your convenience.

The results of this analysis should be part of your Allbase plan (working document).

1. How will I educate my users?

By this time you have already been trained on Allbase and have (or should have) some experience with this product. I would consider some of the options mentioned in the "Get Education" section above. If your company is large enough to have an Education Department, get them involved. Otherwise, contact HP to determine what options would work for your company.

2. How will I handle problem resolution?

This will depend largely on the support contract you have with HP. One approach is to have an on-sight expert (most likely you) that would handle problems as they arise and obtain help from the HP Response Center. [Author note: the HP Response Center, is one of the best when it comes to relational database support.] Another help would be to keep in touch with other Allbase users both within and outside your company (attending conferences is great way to find other users).

3. Who will provide logical and physical database support?

If you are from a small company, I would consider contracting some outside consulting services for at least logical database design. Most likely you will be the one doing the physical design. Be sure to have HP's latest manuals for database administration and performance.

4. What happens if performance is bad?

Any database must be fine tuned once it gets into production. You will never be able to predict all the variables that will impact performance in advance. Most of the information is contained in HP's manuals for Allbase. However, you may require some of HP's consulting assistance until you get up to speed. HP has a tool called SQLMON that can be very helpful (documentation for SQLMON was not available at the time that this paper was written).

5. What tools do we need?

There are two types of tools to consider with this question: database tools and access tools. A listing of tools is included in the Appendix below.

Database tools include software for investigating and manipulating your existing database environment. I recommend doing a product trial with the pilot Allbase project mentioned above. This will reveal whether the software that comes with Allbase (ISQL, SQLGEN, SQLMON and SQLUTIL) will be sufficient to meet your needs.

In addition, your users will require some access tools. Either you or someone on your support team should evaluate some potential products. I highly recommend putting together a team of end users for this evaluation and letting them decide what best meets their requirements.

6. Are there site standards?

Your site should have standards for development and production that must be taken into account. Areas that would impact Allbase support include backup, recovery, programming language support and tools support.

7. How will we migrate to Allbase?

Migration can be an expensive proposition. A lot depends on the type of migration. If you are moving an application to an HP-UX platform, you most likely will be rewriting your application. If you are moving from Image/SQL (TurboIMAGE) to Allbase you may wish to look into migration tools or redesigning your application. Migration is covered in more detail below.

8. How can I keep others interested?

Once you introduce Allbase at your company, you need to keep your current and potential users informed on a regular basis. Use status reports to keep your management informed. Use internal newsletters to communicate with your users. I highly recommend having meetings periodically with your users to demonstrate tools and applications. Ask your HP sales representatives to include Allbase in their presentations to users.

Consider Support -- Multiple Sites

If you only have a single HP site you may skip to the "Migration Considerations" section below.

Multiple site companies have the advantage of numbers. This presents some additional support questions.

9. Could we have our own internal user group?

With nearly 90 separate HP computer sites, 3M has established an internal user group that meets quarterly. We have coordinated our meeting times and days when most of our HP users will be at our corporate headquarters. Those that can't travel may attend via telephone conference. We have found this to be an excellent means of keeping our users informed and interested in Allbase even though many have not yet developed their first relational database application.

10. How will we handle multi-site application support?

At 3M, we have common applications that run at several sites. When it comes to support, who does what kind of support is especially important to our end users. A lot depends on the application lead and the capabilities of his/her team for support. I will negotiate with the application lead as to what level of support I will provide. In some cases, I will provide only technical support that covers performance, consultation and problems with Allbase itself. In other cases, my support will include physical design and structural maintenance (besides technical support). Make sure that your end users know who to call when they experience problems, this interface should be the same no matter what application they have problems with.

11. Who should the end users call for help?

The answer to this question will depend on the support contract you have with HP. In some cases, end users will be valid callers directly to the HP Response Center. With multiple sites, it may be more economical to set up an internal help line for both business and non-business hours.

12. Can we get outside help?

ISVs (independent software vendors) of database tools and access tools can provide additional support that would reduce the demand on internal support at no extra cost for their products. This should be a factor in choosing a tool for your company. Some vendors have 24-hour help available.

13. What about interoperability?

With multiple sites, interaction between these sites will grow as technology makes it possible to do so. At 3M, we believe that client-server is the best solution for meeting the information requirements for our business. With Allbase, the capability for distributed database or replication of databases can provide solutions for your requirements. A key issue is the compatibility of the tools that is used at your company. Hardware and software standards are the keys for interoperability.

Migration Considerations

Now that you have chosen relational database as your standard for data access, the next question is how to get there? Cost will be the key constraint for migrating your data to Allbase. There are several approaches and variations of approaches that must be considered.

1. Purchase off-the-shelf applications

If you can buy the software that would satisfy your business requirements, you can save a lot of internal resources over the life of the application. The vendor becomes a key partner in your support strategy. Some vendors will supply resources for the migration of your data.

2. Rewrite existing applications

When an application gets to the end of its useful life and employs old technologies that are being phased out, it's time to consider rewriting it. Most of my internal users are considering this approach. They have considered where they want to be five years from now and are developing new applications to replace old ones to fit their plan.

3. Convert existing applications

Some TurboIMAGE applications are well suited to conversion. There may not be much need for change in the software itself if it performs its function adequately. If relational data access becomes a requirement there are two options: Image/SQL and DataOne. As Image/SQL rolls out, the upgrade is automatic. With ODBC (Open Database Connectivity) coming soon, most tools will work with Image/SQL. If performance is an issue and your database requires heavy read/write access, conversion to Allbase will increase your options. Pantechnic's DataOne product can convert your database to Allbase without having to change your TurboIMAGE calls. Then it becomes a matter of fine tuning to meet your performance requirements.

4. Create one function at a time

When you model your data, you create "entities". Examples of data entities include: department, employee, etc. Data "elements" are then mapped to entities that translate into tables in your database. When converting applications, you can save a lot of resources (and pain) by converting one function at a time with its associated entities.

5. Build a data warehouse

Data warehousing has become another approach to developing relational data access. Today we have an enormous investment in legacy systems. Conversion of these applications is not economically feasible. By re engineering legacy data into a relational database, it becomes more available for ad-hoc query. The data would be refreshed weekly, daily or more often if required.

Appendix: Allbase Tools

DBMS Software

For the purposes of this paper, I have classified tools for Allbase into two main categories: DBMS software and access software. DBMS software consists of those tools that deals with the physical database environment.

The following tables list the database tools that were known as of June 1992 and since then many more tools have been introduced. This tool set will expand as ODBC is released.

Tool	Vendor
Environment Enhancement	
Allbase/DB2 Connect	HP
Allbase/Net	HP
Allbase/Replicate	HP
Allbase/Star	HP
Environment Migration	
Image/SQL	HP
DataOne	Pantechnic
Structure Utilities	
DBGENERAL/SQL	Bradmark
FLEXIBASE/SQL	Proactive Systems
Indexing Utility	
OMNIDEX	DISC
Documentation Utility	

SQLPaint	Hermes Softlab

Access Software

The following table contains tools for the retrieval and manipulation of data from/to an Allbase Environment. The number of available tools will explode with the introduction of the Allbase PC API ODBC interface.

Tool	**Vendor**
4GL Toolset	
Allbase/4GL	HP
FOCUS	Information Builders
PowerHouse	Cognos
Speedware	Speedware Corp.
4GL Report Writer	
Allbase/BRW	HP
ASKPLUS	Vital Soft
DBA/QUERY	Lee Tech Software
4GL Information Manager	
Prophet/XL	Pantechnic
CASE Toolset	
PowerHouse	Cognos
Speedware	Speedware Corp.
UDMS	Interactive Software Systems
PC-based CASE	
Ingres 4GL	ASK/Ingres
PowerBuilder	Powersoft
Natural Language	
DBfree	Los Altos Software
PC-based Query	
Impromptu	Cognos
Information Access	HP
NewWave Access	HP
SQL/Windows	Gupta Technologies
Window add-on	
OmniWindow	DISC
Client Server	
PowerHouse Windows	Cognos

#5020

I'M CONFUSED...WHAT'S THE DIFFERENCE BETWEEN IMAGE & RELATIONAL

Submitted By:

C. BRADLEY TASHENBERG

From 1975 to 1988 no one in the HP 3000 community had a concern as to where they stood. There was MPE and there was IMAGE. There was a stable environment and there were 50,000 reasonably happy customers. Granted, there was some concern about the high-end limit on the Series 70 until the Spectrum came along. But the Spectrum more than solved that problem. However, there was no confusion as to how an HP 3000 user should proceed in life. Then, around 1988 HP announced a new database technology called Allbase.

Originally, Allbase was supposed to consist of HP-IMAGE (which was not IMAGE) and SQL. However, as time moved on and customer feedback took its toll, HP-IMAGE was replaced by TurboIMAGE and Allbase was trimmed down to only represent SQL-- which became Allbase/SQL. What's unfortunate is that along the way the original concept of what Allbase was supposed to represent was lost. Looking back at the original structure of Allbase, what you had was the consolidation of all known database technologies placed under a common umbrella: HP-IMAGE representing the hierarchial and network structures, and SQL representing the relational model. Therefore, for the HP 3000 users who wanted to use a relational model, HP offered SQL (the industry standard relational database). But for the user who wanted to stay with the network model, an enhanced version of IMAGE was available.

So what happened? Why didn't HP promote Allbase as originally intended, simply replacing HP-IMAGE with TurboIMAGE? It made all the sense in the world; why didn't it happen?

The following is pure conjecture on my part. Back in the 80's, HP went to the Gartner Group, one of the most widely reputed market research firms in the computer industry, and asked them to perform a study to predict the database trend for the 90's. What Gartner presented HP in response was flabbergasting. Gartner stated that although only 7% of the commercial computer market used relational databases in 1988, they predicted that by 1992 over 65% of the commercial market would be using them. To be fair to the Gartner people, this was probably predicated on the success of Oracle and Ingres in the late 80's, and the success of the relational bases in the PC market (of which there are 25,000,000 systems). In 1990, Oracle made $1 billion in sales. This was unheard of for a software company prior to that time. And following on the coat tails of the success of Oracle were companies like Sybase, Informix and others, who were growing equally in interest.

There is no doubt that the interest was there in the mid to late 80's when Gartner first performed its review. Even in the PC market, DBase and Paradox were heavily in vogue. However, by the end of the 80's things changed. The two biggest companies were no longer in a growth role. Oracle's growth was falling off and the company was closing offices and cutting back. Ingres almost went bankrupt before it was sold off to ASK. And DB2, though it was talked about, had very few production installations.

The rage to move to relational was cooling by 1990, but I don't think Gartner or HP had realized that. Articles appeared in both Byte and PC magazines citing the limitations of relational bases in real world applications. However, HP accepted Gartner's analysis and started heavily promoting their own and other relational databases, in the hope of jumping on what they perceived as the relational bandwagon. Why else would they have invited Oracle, Ingres, Sybase, and Informix into the HP marketplace. Certainly not to invite competition. Therefore, the only reasonable conclusion is that HP was convinced that this was the direction of the database interest, and, unless they had a relational database on the HP 3000, they would be in trouble in the early 90's.

Therefore, at the 1990 Interex International show in Boston, HP through Rolandt and Murphy, took the podium and promoted Allbase as HP's future database solution--to the point that the current IMAGE users felt that HP had abandon IMAGE. Pandemonium ensued! The end result was the re-creation of SIGIMAGE, an organization that had been dead for several years. The controversy was so great that the SIGIMAGE meeting, originally budgeted for one hour, took 2 hours and reconvened the following day for another 2 hours. Approximately 200 very irate customers, representing many of the major vendors and large HP customers, banded together to let HP know of the mistake that they were making.

What was unfortunate is that something that could have been perceived by the HP community as a sign that HP was committed to offering new technologies to its customers, was perceived, instead, as pulling the rug from under its customers. If we were Machiavellian, one would think that someone in the HP marketing organization was paid to sabotage HP. Otherwise, why would a vendor with a loyal customer base jeopardize losing that base by recommending that the customers move to an industry standard technology (where there is no vendor protection) from a proprietary technology such as IMAGE. It makes no sense. Every vendor wants to lock its customers in, and that is what proprietary operating systems and proprietary databases do.

So why did HP do what it did? My personal feeling is that HP put too much confidence in its market research firm and not enough confidence in its loyal customer base. Had they only polled a cross section of their customers, they would have quickly realized that there was no need for concern. The majority of the customers loved IMAGE and had little or no interest in relational technology. Even Orly Larson, HP's database spokesperson, could have told them that. At the end of each of his "relational database" speeches he asked how many people were interested in looking at Allbase or SQL. The results were dismal.

Now, two years down line, HP has realized the success of their IMAGE product and has demonstrated tremendous support for the future of IMAGE. HP has guaranteed the HP 3000 community that IMAGE will be supported at least up to the year 2000. The IMAGE lab is healthy and well, the projects are numerous and the responsiveness of HP to user requests couldn't be better. Even the infamous "critical item update" request has not only been addressed, but is coming out in the next release. I understand it's being heavily tested in Sun Valley, Idaho.

The other frequently requested enhancement, "partial key lookup", is also being addressed, but in a different way. Though IMAGE is excellent in quickly retrieving information based on a designated key, it lacks the flexibility to easily retrieve information based on portions of the key. For example, to find information on a customer within an IMAGE Master dataset, you have to know the customer number, or you are not able to access the information. However, if you could build an index that consisted of all customer names, sequentially sorted, and use that as a keying mechanism into the set, then you wouldn't need to know the customer number, or even the exact spelling of the customer's name. A portion of the name could be sufficient. This is the underlying concept of "partial key lookup".

If this is such a great feature, then the question is, why didn't IMAGE have it to begin with? Back when IMAGE was first developed, the model under which it was designed was based on the technology of the most widely accepted network database product of the time. The time was 1972, the product was Cincom's TOTAL. Back then, the Indexed Sequential Access Method (ISAM) was the most popular method of randomly accessing information using a key value. Though ISAM didn't directly support partial key lookups, it did have the internal structure to do so. However, ISAM required unique keys and most of the lookups were through the use of the exact key.

ISAM in its original design was maintenance prone. When you first established an ISAM file, the information was actually spread between two files: an index file and a data file. The data file was randomly maintained whereas the index file was sequentially organized. The index file was further divided in two sections: a prime data area and an overflow area. The prime data area consisted of the originally sorted keys, whereas the overflow area was used to add new keys. Because of this design, periodically, based on the amount of transactional activity of the file, the index had to be reorganized.

Online processing was just coming of age during the early 70's. In reviewing the ISAM approach, Cincom thought that for online access they could replace the index file by a mathematical algorithm, as is used in both TOTAL and IMAGE. The advantage of this approach was that it eliminated the need for an index file entirely, as well as the periodic requirement of the index reorganizations. The disadvantage, of course, was that nothing going into the file was in sorted order. Therefore, you couldn't read your information sequentially, only serially; and that's not the same. In order to produce sequentially ordered reports, you had to extract the information and sort it, prior to reporting it; what was saved in the online environment, impacted the batch processing environment. So when reports had to be run, time had to be available to extract and sort the information, and space was needed to hold the sorted output. So, in practicality, if you did frequent reporting, there was little or no advantage to this algorithmic approach.

When HP developed IMAGE, they chose to go with the algorithmic method established by Cincom and even used by IBM in their new IMS database product. However, looking at the resulting IMAGE design, one of the biggest weaknesses in IMAGE is the lack of indexing. This fault came out very early, back when the IBM System/3 customers were trying to find a similar technology to the ISAM structures that they were using. The answer, of course, was the development of KSAM. However, KSAM never got integrated into IMAGE. Yet one of the most frequently requested enhancements to IMAGE still remains the partial key lookup.

Well, after 15 years of waiting, HP has finally resolved this issue, in an unprecedented way. For the first time that I am aware, HP has allowed third party vendors, who have successfully developed the indexing technology, to be invited into the internals of IMAGE. Its called the "TPI" (Third Party Interface) and it works like this. In version 4.0 of IMAGE, you will see new modes appearing in various IMAGE intrinsics (primarily in the DBOPEN, DBFIND, and DBGET intrinsics). These new modes will allow the IMAGE user to link directly into either of the two third party indexing products and link as if it were an extension of IMAGE. Fantastic!

Two things have come out of this. One, the user will finally have the ability to perform partial key lookups directly from IMAGE, as they always wanted. Two, HP has demonstrated its commitment to work closely with its third parties as valuable business partners. Granted, this is not a free technology. The user will still have to purchase Superdex (or that other product).

However, there's a third benefit that might easily be missed in the use of this indexing technology, and that is the ability to perform relational access within an IMAGE or network database environment. Many people don't realize that "indexing" is the engine behind the relational model (i.e. SQL). Well, now you can have it without ever leaving IMAGE. In the case of Superdex, you even have it before the TPI comes out. And, the advantages are numerous. There is no conversion, no significant reprogramming, no major retraining, and, best of all, no great cost. I would not be surprised that as an added benefit, HP is expands the Allbase TurboConnect to SQL so that future versions of SQL are fully supported through IMAGE (with indexing as the driving engine).

After 3 years of confusion, finally there seems to be some method to the madness. For those companies that are multi-platform oriented and are concerned with preserving an "open systems" standard, Allbase/SQL, Oracle, Ingres, Sybase, or

Informix may well be the best way to go. However, for those of us that are happy with IMAGE but would like to be able to perform relational access within an IMAGE environment, then indexing tools (and the upcoming integration of them into IMAGE) may very well be the way to go.

In the next year, all the options will be at your disposal. If you are happy with IMAGE and don't want to do anything. Great! IMAGE will support you until, and beyond, the year 2000. For those that want partial key access, but only through IMAGE, the TPI will be there at the end of this year. If you want relational access and don't want to abandon your investment in IMAGE, but don't want to wait until the TPI comes out, the third party indexing tools will achieve that for you, and at a minimal cost. And, if you want to try SQL but don't want to make the commitment to convert your IMAGE base over to the Allbase structure, then Turbo Connect may be just for you.

So in the final analysis, things aren't so bad; where there were essentially no choices before, there are now almost too many choices. But alternatives aren't bad. So look at the alternatives. You might just happen to like one of them.

Interex '93 Conference and Expo
San Francisco, California
Paper Number 5021

101 (More Or Less) Moral Things To Do With HPSusan & The Other MPE/iX Predefined Variables In The Harem

David L. Largent

N.G. Gilbert Corporation
700 South Council
P.O. Box 1032
Muncie, Indiana 47308-1032
U.S.A.

Telephone 317/284-4461 Facsimile 317/288-2079

Copyright © 1993 David L. Largent

1 *Introduction – "Is this for me?"*

Have you been looking for a way to make your job streams "smarter"? Would a way to automate some of your procedures be helpful? Would you like to develop a menu to select programs and commands from – without using a programming language? Have you ever wanted to know exactly who was running a program, or how long they had been logged on? If you answered "yes" to any of these questions (and are using an MPE XL/iX system), and have never considered using the MPE/iX Predefined Variables, now is the time to do so. Add some spice to your life!

On the following pages will be found answers to questions relating to MPE/iX Predefined Variables such as:

- ☞ What are they?
- ☞ How do they differ from JCWs?
- ☞ How can they help me?

☞ How are they used?
☞ If I ask them nicely, what will they tell me?

To find the answers, we will examine what the predefined variables are and how they are used. We will examine a number of example command files, job streams, UDCs, and programs to see how and where MPE/iX predefined variables can be used and how they work. It is expected that you will find *at least* one idea (likely more) to put to use at your office. The information presented here is correct and up-to-date (to the best of my knowledge) and reflects the way predefined variables are used and work as of the B.40.00 release of MPE/iX (4.0).

This paper assumes you have a general knowledge of the HP 3000; specifically, the following are "prerequisites" (in varying degrees):

✓ Knowledge of HP 3000 accounting structure (i.e., File/Group/Account/User).
✓ Knowledge of how to use an ASCII text editor (e.g., EDIT/3000).
✓ Knowledge of many MPE/iX system commands and how to use them.
✓ Knowledge of command files and how to use them.
✓ Knowledge of UDCs and how to use them.
✓ Knowledge of job streams and how to use them.

This paper is directed at new users of an HP 3000 MPE/iX computer system who have not yet explored the world of MPE/iX predefined variables. However, *any* user may find some tidbits to put in his or her "tool box" that will prove useful either now or in the future. So, for a good time... check out HPSusan!

2 *What are MPE/iX predefined variables?*

A predefined variable is one way MPE/iX permits programs and commands to communicate with each other within a given job or session. Predefined variables are used at the operating system level and can take on virtually any value. Each predefined variable has a name and can be set and/or interrogated either by the operating system, MPE/iX system commands, or programs.

2.1 *A brief comparison of JCWs and session-level variables.*

An extremely close relative (adopted child?) of the session-level variable is the JCW. JCWs are actually a subset of session-level variables, being restricted by values and naming conventions. The MPE V idea of providing users with operating system-level variables was expanded in MPE XL (and later in MPE/iX) to include data types other than numeric. MPE/iX session-level variables can retain data in Boolean form, 32-bit numeric form, and string form. JCWs are able to retain information in 16-bit numeric form only.

These new variable types have made it possible for MPE/iX to provide a wealth of new information not readily available to the user previously. The information in the predefined variables range from the formatted time of day and job count, to the interactive state and CPU time used. Other variables are able to control the user's environment, such as changing the MPE/iX prompt and automatically logging the user off if there has been no activity within a given amount of time. In addition to the predefined variables, you may also create your own variables.

A further discussion of JCWs and user-defined variables is outside the scope of this paper. As we progress through the material, however, there will be occasional references to user-defined variables. *I strongly encourage you to learn all you can from other sources about the full range of variables.* They will open up many doors you may not even know you want open!

2.2 *How can predefined variables help me?*

Good question! Properly used, predefined variables will permit you to create "smarter" job streams. They can help to automate some of the decision making process in procedures. They can even help you catch errors before they become a problem! All of this is to say: predefined variables can help make the system more effective.

Predefined variables can be checked to determine if certain events have occurred within MPE/iX. By testing predefined variables against specific values, the user can program conditional statements that take action(s) based on the results of the test.

2.3 *OK, I think I see how they could help me. So, how do I use a predefined variable?*

First, some background information. Session-level variables are stored, along with JCWs, in the session's Variable Object Table. The Job Entry Table contains a pointer to this table, and is managed by MPE/iX. This table is shared by all processes in a given job or session. Two classes of variables exist: predefined and user-defined. In some ways, they are exactly the same – in other ways, they are completely different.

Predefined variables are named and assigned values by the system. Some of them may also be assigned values by the user. Both the system and the user may interrogate predefined variables. User-defined variables are named and assigned values solely by the user. MPE/iX never changes the value of or interrogates a user-defined variable. This class of variables will not be discussed at length in this paper.

2.3.1 Predefined variable usage in jobs and/or sessions.

OK, so now you have a flavor of what the predefined variables are. But, how do you look at or set their values? For jobs or sessions, the answer involves the SHOWVAR, and SETVAR system commands. These commands correspond to the MPE/iX system commands SHOWJCW and SETJCW used for JCWs. (MPE/iX distinguishes between variables created by the SETJCW command and those created by the SETVAR command by way of an internal "flag".) Two additional commands, INPUT and ERRCLEAR, may also be used to manipulate predefined variables.

2.3.1.1 The SHOWVAR command.

The SHOWVAR command displays the name(s) and current value(s) of one or more variables. Its syntax is:

```
SHOWVAR [varname][,varname] ... [,varname]
```

where *varname* is a valid variable name (any class). If a name is provided, then only the name(s) and value(s) for the provided variable(s) will be displayed. If a name is not provided, then all user-defined variables and JCWs and their values are displayed – predefined variables are not displayed. You may use the standard wildcard characters to specify a set or range of variables. To display information for all JCWs and variables – including predefined variables – use @ as the *varname*. This command can be executed from a session, job, program, or in BREAK. It is BREAKable (it aborts execution of the command).

For example, if you wish to see the current value of a specific predefined variable, you might type:

```
SHOWVAR HPJOBNAME
```

and the system would respond with:

```
HPJOBNAME = DAVE
```

2.3.1.2 The SETVAR command.

Big deal, so you can look at the value of a predefined variable. So what?!? OK, let me tell you how you can change or set the value of some of the predefined variables – that is a little more productive. We need the SETVAR command to do this:

```
                   {<space>}
SETVAR varname     {,      }   expression
                   {;      }
```

where *varname* is the name of a new or existing user-defined or predefined variable and *expression* represents a value, variable or JCW name, or an expression that will be evaluated and assigned to the specified variable. This command can be executed from a session, job, program, or in BREAK. Pressing BREAK has no effect on this command.

A value in *expression* may be Boolean (true or false), integer (hexadecimal, octal, or decimal), or string (quoted). For a list and explanation of the operators allowed in *expression*, see the "Commands" reference manual. The JCW mnemonics (discussed in a later section) are also allowed.

When the SETVAR command is executed, it causes the MPE/iX Variable Object Table for your job or session to be scanned for the name of the specified variable. If the name is found, the variable is set to the value provided. If the name is not found, it is added to the table and then set to the value provided. Once a variable is created, it exists for the duration of that session or job, or until the DELETEVAR command or the HPCIDELETEVAR intrinsic is used to delete it.

For example, to set the predefined variable called HPREDOSIZE to a value of 50, you could type:

```
SETVAR HPREDOSIZE 50
```

And finally, if you want to create a user-defined variable that has a value representing two days from now, you could type:

```
SETVAR TWODAYSFROMNOW HPDAY+2
```

This will take the current value from the predefined variable HPDAY and add two. Obviously you would need to follow the statement with an "IF" statement to handle the end of the week, but I think you probably get the idea.

2.3.1.3 The INPUT command.

Another way of setting a value for a variable is by using the INPUT command. This command permits the interactive setting of a variable based on input from a user, and would most typically be used in a command file or UDC. Its syntax is:

```
INPUT [NAME=]varname [;PROMPT=prompt] [;WAIT=seconds]
```

where *varname* is the name of a new or existing user-defined or predefined variable which could be set with SETVAR. If the prompt parameter is specified, then *prompt* is displayed on the standard listing device ($STDLIST) before accepting input from the user. If delimiters (for example, comma or semicolon) are part of *prompt*, you must enclose the entire prompt string within quotation marks (" or '). The default is that no prompt is displayed. The *seconds* parameter specifies how many seconds MPE/iX will wait before aborting the command. The default is zero seconds, which means no time limit. This command can be executed from a session, job, program, or in BREAK. It is BREAKable (it aborts execution of the command without creating or modifying *varname*).

When the INPUT command is executed, *prompt* is displayed (if specified) on $STDLIST before a value is read from the user. A read is then posted to the standard input device ($STDIN). This is a timed read if the *seconds* parameter is specified, and will terminate when either the user provides input, or the timed read expires. The INPUT command will terminate (and the session will be logged off) if the number of minutes specified in HPTIMEOUT passes. Once a value is provided by the user, the operation of the INPUT command is comparable to the SETVAR command. There are two exceptions, however: INPUT stores the user's value as a string (performing no expression evaluation), and INPUT's behavior when a timed read terminates.

The INPUT command does not evaluate expressions, but simply stores the user's response as a character string. If you want the user's response to be treated as an expression and evaluated, you must follow the INPUT command with a SETVAR command as shown below.

```
INPUT JUNK
SETVAR JUNK !JUNK
```

Note that if the expression involves character strings, and the user does not enclose them in quotes, the execution of the SETVAR command will result in an error.

If the user responds to an INPUT command by pressing the return key (no other input), and the variable previously existed, the existing value is not changed. New variables are set to null ("") given this same situation. If a timed read expires before the user responds, values are treated the same as if the user had simply pressed the return key. In this situation however, a warning occurs and CIERROR is set to 9003.

Something to note is that if a colon (:) is read (as the first character) by the INPUT command at any level other than the root level of the command interpreter (CI), the following error message is returned:

```
END OF FILE ON INPUT (CIERR900)
```

For example, to prompt the user for a value for the predefined variable called HPREDOSIZE, you could use the following command in a UDC or command file:

```
INPUT HPREDOSIZE,"Enter the new value for HPRedoSize",10
```

This will display the prompt "Enter the new value for HPRedoSize" on $STDLIST, and wait 10 seconds for the user's response. If the user does not respond within 10 seconds, the value of HPREDOSIZE will be left unchanged.

2.3.1.4 The ERRCLEAR command.

A command that becomes useful as you develop command files, UDCs, and/or job stream files is ERRCLEAR. It initializes (zeros) all HP predefined error-related variables. Its syntax is rather simple:

```
ERRCLEAR
```

That's right – there are no parameters! This command can be executed from a session or job. It is not BREAKable. Issuing the ERRCLEAR command is equivalent to issuing the following commands:

```
SETVAR CIERROR OK
SETVAR HPCIERR OK
SETVAR HPCIERRCOL OK
SETVAR HPFSERR OK
```

You are still not feeling very productive with predefined variables yet, right!?! OK, here is the good stuff. Predefined variables are most often used to control the flow of batch jobs (they can also be used in command files, UDCs, and/or in sessions), taking various actions based on the results of previous steps. To do this, the IF/THEN, ELSE, ELSEIF, ENDIF, CONTINUE, ESCAPE, EXIT, RETURN, WHILE and ENDWHILE commands are used. For purposes of this paper, I am going to assume you either know how these MPE/iX system commands work or can easily acquire the knowledge as we look at examples. (Some examples will come later that should clear up some of your questions.)

2.3.1.5 The JCW mnemonics.

A word or two about the four JCW mnemonics. These can be used to assign values to variables, or used in IF commands. They are "defined" in the table below.

JCW Mnemonics

Mnemonic	Value
OK	Zero
WARN	16,384
FATAL	32,768
SYSTEM	49,152

These are strictly mnemonics for specific values – they cannot be used as variable or JCW names. You may use a combination of a mnemonic and a number to indicate a value between two mnemonics. If you specify:

```
FATAL32
```

for example, an implied addition takes place (32,768 + 32) and the value would be 32,800. The "+" and "-" option can also be used with mnemonics. For example:

```
FATAL - 768
```

will result in a value of 32,000.

2.3.2 *Predefined variable usage in programs.*

Now, for you programmer-type people, we will look at how to interrogate, and set predefined variables from within a program. The intrinsics HPCIGETVAR, and HPCIPUTVAR will be utilized. These correspond to the MPE/iX CI commands SHOWVAR and SETVAR used on standard variables. (Actually, these variable intrinsics may be used to set and display JCWs as well.) MPE/iX distinguishes between variables created by the PUTJCW intrinsic and those created by the HPCIPUTVAR intrinsic by way of an internal "flag". My examples are based on COBOL II/XL usage; however, I will try to provide information in a general way.

2.3.2.1 *The HPCIGETVAR intrinsic.*

The programmatic equivalent to the SHOWVAR command is the HPCIGETVAR intrinsic. Its syntax (in COBOL II/XL format) is:

```
CALL INTRINSIC "HPCIGETVAR" USING varname,status [,itemnum,item] [ ... ]
```

where *varname* is an alphanumeric variable (character array) containing the name of the variable (or JCW) to be found, and *status* is a signed 32-bit integer variable to which a value denoting the execution status of the intrinsic is returned. The *varname* parameter may contain up to 255 alphanumeric characters, starting with a letter (or the underscore (_) character) and ending with a non-alphanumeric character, such as a blank. (The underscore is not a valid delimiter since it is a valid variable name character.) The name is not case sensitive. If the requested variable is found in the session's Variable Object Table, its value is returned to the program in an *item* parameter; if not found, no change is made to this parameter. The *status* parameter will be returned with one of a number of possible values.

If no errors or warnings are encountered, all 32 bits of the *status* parameter are set to zero. If an error or warning is encountered, the first 16 bits (0-15) provide an error or warning number. Error conditions are represented by negative values; warnings are represented by positive values. In an error or warning situation, the second set of 16 bits (16-31) will be set to 166 which is the subsystem identifier for HPCIGETVAR. Shown in the table below is a partial list of possible error numbers that may be returned. This list was obtained from an old edition of the *MPE XL Intrinsics Reference Manual*. The current edition does not specifically list the errors (nor are they documented anywhere else!). I am sure the list is neither complete nor up to date.

A Few Status Parameter Values for the HPCIGETVAR Intrinsic
(First Word Only)

Status Value	Meaning or Result
0	Successful execution: The value of *varname* has been retrieved from the session's Variable Object Table.
-8106	Error: *Varname* was not found in the session's Variable Object Table.
-8110	Error: *varname* is not valid.
-8156	Error: *varname* is longer than 255 characters.

The *itemnum* parameter is an unsigned 32-bit integer variable in which you specify the item you want retrieved. See the table below for possible values. The *item* parameter varies in type (see the table below), and is used to return the requested information or value. The HPCIGETVAR intrinsic can be used to obtain the value and/or attribute of any of the classes of variables or JCWs. You may specify up to six *itemnum/item* pairs.

Item Number/Item Parameter Values for the HPCIGETVAR Intrinsic

Item Number	Item Type	Description or Meaning
0	(not applicable)	*Itemnum/item* pair ignored.
1	32-bit signed integer	Integer value of *varname*. Zero is returned if *varname* is not an integer variable.
2	Alphanumeric (character array)	String value of *varname*. An ASCII zero is returned if *varname* is not a string variable. Item pair 10 should also be used with the pair.
3	32-bit signed integer	Boolean value of *varname*, where true = one, and false = zero. Zero is returned if *varname* is not a boolean variable.
10	32-bit signed integer	Length of array (character string) passed to hold *varname*'s string value. If length is passed and an array is not, an error occurs. The default length is 255. Used in conjunction with item pair 2.
11	32-bit signed integer	Actual length (in bytes) of *varname*'s string value.
12	32-bit signed integer	Value of *varname*: nonzero if *varname* is recursively dereferenced; zero for a level-1 value. Default value is one (nonzero).
13	32-bit signed integer	Type of variable, where integer = one, string = two, and boolean = three.

To accomplish the same task in RPG/XL, use the INTR operation to call HPCIGETVAR. If you are not familiar with how to do this, you may want to reference *HP 3000 Application Note # 94: RPG/XL Intrinsic Interface* dated January 15, 1992 (document part number 5960-8223).

2.3.2.2 *The HPCIPUTVAR intrinsic.*

The SETVAR command's programmatic equivalent is the HPCIPUTVAR intrinsic. Its syntax (in COBOL II/XL format) is:

```
CALL INTRINSIC "HPCIPUTVAR" USING varname,status [,itemnum,item] [ ... ]
```

where *varname* is an alphanumeric variable (character array) containing the name of the variable (or JCW) to be created or changed, and *status* is a signed 32-bit integer variable to which a value denoting the execution status of the intrinsic is returned. The *varname* parameter may contain up to 255 alphanumeric characters, starting with a letter (or the underscore (_) character) and ending with a non-alphanumeric character, such as a blank. (The underscore is not a valid delimiter since it is a valid variable name character.) The name is not case sensitive. If the requested variable (or JCW) is found in the session's Variable Object Table, its value is changed to the value provided in an *item* parameter; if not found, an entry is created and then is assigned the specified value. The *status* parameter will be returned with one of a number of possible values.

If no errors or warnings are encountered, all 32 bits of the *status* parameter are set to zero. If an error or warning is encountered, the first 16 bits (0-15) provide an error or warning number. Error conditions are represented by negative values; warnings are represented by positive values. In an error or warning situation, the second set of 16 bits (16-31) will be set to 166 which is the subsystem identifier for HPCIPUTVAR. Shown in the table below is a partial list of possible error numbers that may be returned. This list was obtained from an old edition of the *MPE XL Intrinsics Reference Manual*. The current edition does not specifically list the errors (nor are they documented anywhere else!). I am sure the list is neither complete nor up to date.

A Few Status Parameter Values for the HPCIPUTVAR Intrinsic
(First Word Only)

Status Value	Meaning or Result
0	Successful execution: *varname* has been entered in the session's Variable Object Table.
-8110	Error: *varname* is not valid.
-8156	Error: *varname* is longer than 255 characters.

The *itemnum* parameter is an unsigned 32-bit integer variable in which you specify the item you want set. See the table below for possible values. The *item* parameter varies in type (see the table below), and is used to specify the value for *varname* and/or other information. The HPCIPUTVAR intrinsic can be used to set the value of any of the user-defined variables or JCWs, and the user-writable variables or JCWs. You may specify up to three *itemnum/item* pairs.

Item Number/Item Parameter Values for the HPCIPUTVAR Intrinsic

Item Number	Item Type	Description or Meaning
0	(not applicable)	*Itemnum/item* pair ignored.
1	32-bit signed integer	Integer value to be assigned to *varname*. No other pairs are required, and will cause an error if present.
2	Alphanumeric (character array)	String value to be assigned to *varname*. Must also provide item pair number 11.
3	32-bit signed integer	Boolean value to be assigned to *varname*, where true = nonzero, and false = zero. No other pairs are required, and will cause an error if present.
11	32-bit signed integer	Actual length (in bytes) of string to be assigned to *varname*. Used in conjunction with item pair 2.
13	32-bit signed integer	String value will store as sting value if this item has a nonzero value. If this item's value is zero, Boolean and integer interpretation of item two's value will be attempted first.

To accomplish the same task in RPG/XL, use the INTR operation to call HPCIPUTVAR. If you are not familiar with how to do this, you may want to reference *HP 3000 Application Note # 94: RPG/XL Intrinsic Interface* dated January 15, 1992 (document part number 5960-8223).

2.4 *How about some examples?*

Good idea! First, we will look at some examples in batch jobs and sessions, then we will take a look at a couple example programs that use predefined variables. As predefined variables are presented in an example for the first time, we will take a look at what it can tell us. In later examples, you will be on your own to remember what it does – you know, the old "building block" theory of teaching/learning. So sharpen your pencils; here we go...

2.4.1 Batch job and session examples.

A note before we get started. Anytime you see the characters <esc> in this paper, read it as an escape character. That is, for purposes of this paper, I have used the characters <esc> in place of the escape character. However, if you were to type in the UDC or command file, you would need to use the escape character instead. Be careful not to mistake the ESC variable used in some of the files. This actually is a variable that has been set to the value of an escape character.

Here is a command file that will tell a user virtually everything they could ever want to know about their logon.

```
Comment  File: ME
Comment
Comment  Set enhancement variables:
Comment      I = Inverse
Comment      U = Underline
SetVar ENHOFF  chr(27)+"&d@"
SetVar ION  chr(27)+"&dB"
SetVar UON  chr(27)+"&dD"
Echo
Echo     !"ION"   USER INFORMATION & STATUS   !ENHOFF
Echo
Echo
Echo     !"UON"User Identification:!ENHOFF
Echo       Session/Job #:     !HPJOBTYPE!HPJOBNUM
Echo       Job/Session Name:  !HPJOBNAME&
            ![rpt(" ",8-len("!HPJOBNAME"))]   User:     !HPUSER
Echo       Current Group:     !HPGROUP&
            ![rpt(" ",8-len("!HPGROUP"))]   Account: !HPACCOUNT
Echo       Home Group:        !HPHGROUP
Echo       Search Path:       !HPPATH
Echo       LDev In/Out #:     !HPLDEVIN/!HPLDEVLIST
Echo       DTC Port ID #:     !HPDTCPORTID
Echo
Echo
Echo     !"UON"Dates & Times:!ENHOFF
Echo       Current:           !HPDATEF at !HPTIMEF
Echo       Logged on:         !HPINTRODATE at !HPINTROTIME
SetVar ME_HOURS  !HPCONNMINS/60
Echo       Time since logon: !ME_HOURS hours &
            ![!HPCONNMINS - (!ME_HOURS * 60)] minutes
Echo       CPU since logon:  !HPCPUSECS seconds
Echo
Echo
SetVar HPMSGFENCE  2
Input ME_DUMMY, "     Press RETURN to see more information.",20
SetVar HPMSGFENCE  0

.....(continued).....
```

```
.....(continued).....

Echo
Echo
Echo     !"UON"Status Information:!ENHOFF
Echo       Redo Stack Size:  !HPREDOSIZE commands
Echo       Command Number:   !HPCMDNUM
Echo       Nested Commands:  !HPUSERCMDEPTH
Echo       CI Level number:  !HPCIDEPTH
Echo       Command Time Out: !HPTIMEOUT minutes
If HPTYPEAHEAD
  SetVar ME_YES_NO  "Yes"
Else
  SetVar ME_YES_NO  "No"
EndIf
Echo       Type Ahead On?    !ME_YES_NO
If HPQUIET
  SetVar ME_YES_NO  "Yes"
Else
  SetVar ME_YES_NO  "No"
EndIf
Echo       Quiet Mode On?    !ME_YES_NO
If HPINBREAK
  SetVar ME_YES_NO  "Yes"
Else
  SetVar ME_YES_NO  "No"
EndIf
Echo       In Break Mode?    !ME_YES_NO
Echo
Echo
Echo     !"UON"Capabilities:!ENHOFF
Echo       Account:          !HPACCTCAPF
Echo       Group:            !HPGROUPCAPF
Echo       User:             !HPUSERCAPF
DeleteVar ME_@
```

As a command file, this is pretty straightforward. Just a bunch of ECHOs, a few IFs and SETVARs, and an INPUT and DELETEVAR thrown in for good measure. Notice the INPUT will time out after 20 seconds, so if the user does not respond right away, it will move forward automatically.

So what do all of those predefined variables mean? Starting at the top of the command file and working our way down, let us take a look. HPJOBTYPE provides a value of "S" if the logon is a session, and "J" if a job. HPJOBNUM is the job/session number. The job or session name is in HPJOBNAME. You'll find the user, group and account stored in HPUSER, HPGROUP and HPACCOUNT respectively, while HPHGROUP holds the user's home group. The search path is defined in HPPATH. The logical device number (LDEV #) for $STDIN and $STDLIST are provided in HPLDEVIN and HPLDEVLIST. And finally, HPDTCPORTID holds the port ID of the DTC the user's terminal is connected to.

OK. Take a big breath; here we go again. HPDATEF and HPTIMEF provide the current formatted date and time. HPINTRODATE and HPINTROTIME will tell you (formatted) the date and time the user logged on. If you want to know how many minutes have passed since you logged on, check HPCONNMINS. There is also HPCPUSECS that provides the number of seconds the CPU has been doing work for your session. Hang on, we still have a few more to go!

If you are not familiar with the LISTREDO command, you owe it to yourself to start looking. HPREDOSIZE is the variable that controls how big your redo stack is. HPCMDNUM tells you the current command number (i.e. how many commands you have executed since you logged on). HPUSERCMDEPTH provides the answer to the question "How deep am I in nested command files and UDCs?". The number of nested CIs is provided in HPCIDEPTH. The number of minutes allowed for CI reads before it logs you off is controlled by HPTIMEOUT. HPTYPEAHEAD is set to TRUE if the type ahead feature is enabled for your session (this is separate and distinct from Reflection's type ahead). If your session is not accepting messages via the TELL command, HPQUIET will be set to TRUE. And almost lastly, if your session is "in break", HPINBREAK will have a value of TRUE. The account, group and user capabilities are stored in HPACCTCAPF, HPGROUPCAPF and HPUSERCAPF. These are formatted (i.e. the bits are translated into letters). Well, if you survived that, you can handle any of the others – this one was by far the highest concentration of predefined variables in one example.

Here is a counterpart for the ME command file that displays the system's information and status.

```
Comment  File: SYSSTAT
Comment  Set enhancement variables:
Comment       I = Inverse     U = Underline
SetVar ENHOFF  chr(27)+"&d@"
SetVar ION  chr(27)+"&dB"
SetVar UON  chr(27)+"&dD"
Echo
Echo     !"ION"   SYSTEM INFORMATION & STATUS   !ENHOFF
Echo
Echo     !"UON"System Identification:!ENHOFF
Echo       System Name:    !HPSYSNAME
Echo       CPU Model:      !HPCPUNAME
Echo       HP System ID#:  !HPSUSAN
Echo       MPE Version:    !HPVERSION
Echo
Echo     !"UON"Session & Job Counts:!ENHOFF
Echo       Executing Jobs & Sessions:  !HPEXECJOBS
Echo          Sessions:    !HPSESCOUNT
Echo          Jobs:        !HPJOBCOUNT
Echo       Non-executing Jobs:
Echo          Suspended:   !HPSUSPJOBS
Echo          Waiting:     !HPWAITJOBS
Echo          Scheduled:   !HPSCHEDJOBS
Echo       Total Users:    !HPUSERCOUNT
Echo
Echo     !"UON"Limits:!ENHOFF
Echo       Current Limits:    Sessions: !HPSESLIMIT   Jobs: !HPJOBLIMIT
Echo       Max Users Allowed: !HPUSERLIMIT
Echo       Job Fence:         !HPJOBFENCE
Echo       Output Fence:      !HPOUTFENCE
```

It is just a bunch of ECHO commands to display the information, but it does give us a chance to talk about what each of the predefined variables do. So, here we go. HPSYSNAME will contain what ever name you have assigned to your system, and (via SETVAR) placed into the variable. HPCPUNAME, on the other hand, is the name of the computer model (for example, SERIES 922LX). The MPE/ix operating system version is stored in HPVERSION.

If you want to know the number of executing jobs and sessions, look at HPEXECJOBS. If you want a breakdown, HPSESCOUNT and HPJOBCOUNT have your answers. The number of currently suspended, waiting, and scheduled jobs can be found in HPSUSPJOBS, HPWAITJOBS, and HPSCHEDJOBS. HPUSERCOUNT is the number of users (defined as non-OPERATOR.SYS) currently on the system.

And finally some limits. HPSESLIMIT and HPJOBLIMIT provide the current settings of the session and job limits. The maximum number of users allowed on the system is stored in HPUSERLIMIT. The job and out fences can be found in HPJOBFENCE and HPOUTFENCE.

If you were following along in the example, you will have noticed I skipped over one of the predefined variables. HPSUSAN. Ah yes, we finally get to talk about the mysterious "woman" from the title of the paper! HPSUSAN is an "electronic serial number" which, according to HP, no other system has. (Via comments from the grapevine, I think there are a few isolated exceptions that turn up from time to time however.)

The name is actually an acronym for the words "System Unique Serially Assigned Number". So what purpose does it serve? About the same as any serial number does. Software companies have found they can use HPSUSAN as a way to tie a particular copy of their software to a given machine, thus controlling software piracy. They simply request your HPSUSAN and include that in some validation logic in their program that prohibits the running of their software if the HPSUSAN from the system does not match the value in the software.

The following is a modified version of the job stream we use to do our system back ups.

```
!Job BACKUP,OPERATOR.SYS;Pri=CS
!Comment*****************************************************
!Comment                    BACKUPJ                         *
!Comment    This job stream backs-up the system (using      *
!Comment  STORE) onto magnetic tape.  A free disk space     *
!Comment  table is produced, as well as a summary of        *
!Comment  computer use by group by account.                 *
!Comment*****************************************************
!Comment
!Comment    Today's Date:  __/__/__
!Comment
!Comment    Tape Number:   DDS-_____
!Comment
!Comment    Line Printer Hours (LDev 5) ____ (Only on FULL BackUps)
!Comment
!Comment    Line Printer Hours (LDev 6) ____ (Only on FULL BackUps)
!DiscFree A
!DiscFree C
!Report
!SetVar MONDAY  2
!SetVar THURSDAY  MONDAY + 3
!SetVar ION  chr(27)+"&dB"
!SetVar ENHOFF  chr(27)+"&d@"
!If HPDAY = MONDAY or HPDAY = THURSDAY then
!  SetVar BACKUP_IS_FULL  TRUE
!ElseIf finfo("FULLDATE","EXISTS") then
!  SetVar BACKUP_IS_FULL  FALSE
!Else
!  Comment  To force a full back up on other days, (perhaps
!  Comment  because the office was closed on the normal full
!  Comment  back up day), simply PURGE FULLDATE.OPER.SYS.
!  TellOp
!  TellOp  !"ION" Full back up date file (FULLDATE.OPER.SYS) !ENHOFF
!  TellOp  !"ION" is missing.  Performing a Full back up     !ENHOFF
!  TellOp  !"ION" instead of the usual Partial back up.      !ENHOFF
!  TellOp
!  SetVar BACKUP_IS_FULL  TRUE
!EndIf
!If BACKUP_IS_FULL then
!  SetVar BACKUP_DATE_F  ""
!Else
!  SetVar BACKUP_DATE_I  finfo("FULLDATE","INTCREATED")
!  SetVar BACKUP_DATE_C  "!BACKUP_DATE_I"
!  SetVar BACKUP_DATE  str("!BACKUP_DATE_C",5,2)+"/"+&
!                      str("!BACKUP_DATE_C",7,2)+"/"+&
!                      str("!BACKUP_DATE_C",3,2)
!  SetVar BACKUP_DATE_F  ";DATE>="+"!BACKUP_DATE"
!EndIf
!SetVar BACKUP_FILE_SET  "@.@.SYS,@.@.@-@.@.SYS"
!TellOp
!If BACKUP_IS_FULL then
!  TellOp  Today's back up will be a Full back up.
!Else
!  TellOp  Today's back up will copy all files modified
!  TellOp  on or after !BACKUP_DATE.
!EndIf
!TellOp
!TellOp  Please mount a tape for the back up.
!TellOp

.....(continued).....
```

```
.....(continued).....

!If finfo("LASTBACK","EXISTS") and &
!    finfo("LASTBACK","CREATED") = HPDATEF then
!   TellOp!"ION"  It appears that a back up has already been  !ENHOFF
!   TellOp!"ION"  performed today.  Abort the job if you do   !ENHOFF
!   TellOp!"ION"  not want it done a second time.             !ENHOFF
!   TellOp
!EndIf
!File BACKUPT;dev=TAPE
!Store !BACKUP_FILE_SET;*BACKUPT;Directory;Show&
!!BACKUP_DATE_F;Progress=5
!If BACKUP_IS_FULL and HPDAY <> THURSDAY then
!   If finfo("FULLDATE","EXISTS") then
!     Purge FULLDATE
!   EndIf
!   Build FULLDATE
!EndIf
!If finfo("LASTBACK","EXISTS") then
!   Purge LASTBACK
!EndIf
!Build LASTBACK
!TellOp********************************************************
!TellOp     The BackUp has completed.
!TellOp     The BACKUPJ job stream is done.
!TellOp     !"ION"Don't forget to set the limits back up.!ENHOFF
!TellOp********************************************************
!EOJ
```

Notice the creation of variables MONDAY and THURSDAY. They are not necessary, but make the IF commands read a little nicer. Also, ION and ENHOFF are set up to use when the inverse video enhancement is to be turned on and off.

Full back ups should be performed on Monday and Thursday; otherwise, a partial back up is done. Which to do is decided by checking HPDAY, which is a number from one to seven, where one is Sunday. The creation date of file LASTBACK is the last time a back up was performed. The creation date of file FULLDATE is the last time a full back up was performed. Both of these files are built, but contain no data, and as such, do not require any disk space. Their sole purpose in life is to provide the last date an event occurred by way of their creation date.

There are around forty MPE/iX expression evaluator functions that do and/or tell you virtually anything you want. Used in this job stream are two of them.

- finfo(...,"EXISTS") returns a value of TRUE if the file exists, and FALSE if the file does not.

- finfo(...,"INTCREATED") returns the creation date of the file as an integer in the form yyyymmdd.

- finfo(...,"CREATED") returns the formatted creation date of the file.

- str(...) extracts individual characters from a character string variable.

Have you been looking for a way to load information into your terminal's function keys automatically? If you have, keep reading. If you have not, perhaps you want to know why you would want to. Using the function keys to execute commands provides a one or two keystroke execution of commands.

My original source for this information was the "Users' Forum" section of the November 1987 issue of *Interact* magazine. I have since seen a number of other similar approaches, and have successfully used a modification of all the ideas on all of the terminals I have access to (HP 2392A, HP 700/92, and HP Vectra PC using Reflection 1 for Windows). I believe it will work on any "HP terminal" that has the capability of "labeling" the function keys on the screen. There are three parts to the solution: two command files that need to be defined once and stored somewhere for everyone's access, and one (or more) additional command file(s) defined that uses the first two.

The two "system-level" command files include SFK:

```
Parm Key = "1"; Attr = "0"; Head1 = "        "; Head2 = "        "; &
     Function = "                                        "
Comment  File: SFK
Option NoList
Setvar ESC  CHR(27)
Option List
Echo !"ESC"&f!"Attr"a!"Key"k16d![LEN("!Function")]L&
     !Head1!Head2!Function!"ESC"M!"ESC"A
```

and DISPFKEY:

```
Parm Mode = "B"
Comment  File: DISPFKEY
Option NoList
Setvar ESC  CHR(27)
Option List
Echo !"ESC"&j!Mode!"ESC"M!"ESC"A
```

The SFK command file accepts the information for one function key and "loads" that information by causing it to be displayed on the terminal with the ECHO command. Be careful when you type this one in: upper and lowercase makes a difference in how it will execute! The KEY parameter signifies which function key (1 through 8). The ATTR parameter indicates what should happen when the function key is pressed:

Values for the Attribute (ATTR) parameter of the SFK Command File

Value	Action to be Taken
0	NORMAL. The defined string is displayed. To execute it, the user must press the RETURN key.
1	LOCAL ONLY. The defined string is displayed; it can not be executed, however.
2	TRANSMIT. The defined string is displayed and immediately executed.

The HEAD1 and HEAD2 parameters provide values to be placed in the labels on the screen. And finally, the FUNCTION parameter provides the actual character string to be "loaded" into the function key.

The DISPFKEY command file simply causes the function key labels to be displayed on the terminal screen (if they are not already). The "<esc>M<esc>A" sequence in both command files effectively "erases" the lines from the screen as the commands execute. If you would like them left on the screen, that sequence could be left off the end of the line.

What might the command files that use these look like? Here is an example:

```
Comment  File: MAINKEYS
If HPJOBTYPE="S" and HPDUPLICATIVE then
  Xeq SFK 1 2 "        " "  Quad  " "Quad.Pub.CSLXL"
  Xeq SFK 2 0 "        " "  Link  " "Link "
  Xeq SFK 3 2 "  Show  " "  Jobs  " "ShowJob"
  Xeq SFK 4 2 "  List  " "Sp Files" "ListSpF @;Detail"
  Xeq SFK 5 0 "  View  " "  File  " "Print "
  Xeq SFK 6 2 "View Spl" "File (2)" ".Out.HPSpool"
  Xeq SFK 7 0 " Change " " Group  " "ChGroup "
  Xeq SFK 8 2 "  List  " "  Redo  " "ListRedo"
  Xeq DISPFKEY
Else
  Echo  This is a job, thus the function keys were not set.
EndIf
```

It is important to note that two eight-character labels must be provided for each of the function key labels. The IF command checks to make sure the commands are only executed in session mode, since it may make sense to reference a command file such as this one from a logon UDC. HPDUPLICATIVE will have a value of TRUE if the logon is duplicative (i.e. it is a session).

If you want to take this a step further, instead of RUNning each of the programs (either implicitly or explicitly), execute a command file to do so. In the command file for each program, execute commands before and after the RUN command to load the keys for the program about to be run and then reset them afterwards. For example, command file QUAD could be defined as:

```
Comment  File: QUAD
If HPJOBTYPE="S" and HPDUPLICATIVE then
  Xeq SFK 1 2 "          " "  Help  " "Help"
  Xeq SFK 2 2 "          " "        " " "
  Xeq SFK 3 2 "List All" "ON Line " "List A"
  Xeq SFK 4 2 "List All" "Offline " "List A O"
  Xeq SFK 5 2 "PageMode" " First  " "Page First
  Xeq SFK 6 2 "PageMode" "  Last  " "Page Last
  Xeq SFK 7 2 "          " "  Keep  " "Keep "
  Xeq SFK 8 2 "          " "  Exit  " "Exit"
  Xeq DISPFKEY
  Run QUAD.PUB.CSLXL
  XEQ MAINKEYS
Else
  Echo  This is a job. thus the function keys were not set.
  Run QUAD.PUB.CSLXL
EndIf
```

By using this nesting technique, you can set up a "menu" system with only command files and terminal function keys. Neat, huh?!?

Here is another way of implementing a menu system that can shield the user from the MPE/iX prompt (and vice versa). It uses a command file to implement the menu logic.

```
parm WHICHCHOICE = Warn
Option NoList,NoBreak
Comment     File:   MENU
Comment     Author: David L. Largent
Comment     Date:   07/06/93
SetVar HPMSGFENCE  2
SetVar HPAUTOCONT  TRUE
SetVar CHOICE  Warn + 1
SetVar CHOICE  !WHICHCHOICE
SetVar RESPONSE  " "
SetVar ESC  chr(27)
SetVar HPMSGFENCE  0
While CHOICE <> 0 Do
  SetVar HPMSGFENCE  2
  If CHOICE = Warn then
    Echo
    Echo Options for !HPUSER.!HPACCOUNT are:
    Echo      1 - Run Report A
    Echo      2 - Run Report B
    Echo      3 - Run Report C
    Echo      4 - Show System Information & Status
    Echo      5 - Show Information About Your Logon
    Echo      6 - Show Who Is Logged On the System
    Echo      7 - Show Printer Output Pending From Your Logon
    Echo      8 - Access the E-Mail System
    Echo      9 - Access the Command Prompt
    Echo      0 - Logoff the System

.....(continued).....
```

```
.....(continued).....

    SetVar RESPONSE  " "
    SetVar MINUTES_PASSED  0
    SetVar HPMSGFENCE  2
    Echo
    While RESPONSE = " "
      Comment  The following mess places the date and time inside
      Comment  of parenthesis in front of the prompt.
      Comment  It also causes the prompt to be displayed over
      Comment  top of the last one.
      Input RESPONSE, "!["!"ESC"A!"ESC"M" + &
            "(" + rpt("0",2-len("!HPMONTH")) + "!HPMONTH/" + &
            rpt("0",2-len("!HPDATE")) + "!HPDATE/!HPYEAR " + &
            rpt("0",2-len("!HPHOUR")) + "!HPHOUR:" + &
            rpt("0",2-len("!HPMINUTE")) + "!HPMINUTE) " + &
            "Type the number of your selection !HPJOBNAME: "]",60
      If HPCIERR = -9003 then
        SetVar MINUTES_PASSED  MINUTES_PASSED + 1
        If MINUTES_PASSED > HPTIMEOUT
          Echo
          Echo  There has been no response from you during the last &
                !HPTIMEOUT minutes !HPJOBNAME.
          Echo  You will therefore be automatically logged off.
          SetVar RESPONSE  "0"
        EndIf
      EndIf
    EndWhile
    SetVar HPMSGFENCE  0
    If numeric(RESPONSE) then
      SetVar CHOICE  !RESPONSE
    Else
      SetVar CHOICE  Warn
    EndIf
  EndIf
  SetVar HPMSGFENCE  0
  If CHOICE = 1 then
    Run Prog1
  ElseIf CHOICE = 2 then
    Run Prog2
  ElseIf CHOICE = 3 then
    Run Prog3
  ElseIf CHOICE = 4 then
    Xeq SYSSTAT
    Echo
    Input DUMMY, "Press RETURN to display the Menu."
  ElseIf CHOICE = 5 then
    Xeq ME
    Echo
    Input DUMMY, "Press RETURN to display the Menu."
  ElseIf CHOICE = 6 then
    ShowJob ;EXEC
    Echo
    Input DUMMY, "Press RETURN to display the Menu."
  ElseIf CHOICE = 7 then
    ListSpf ;Detail
    Echo
    Input DUMMY, "Press RETURN to display the Menu."
  ElseIf CHOICE = 8 then
    Xeq MAIL

.....(continued).....
```

```
.....(continued).....

  ElseIf CHOICE = 9 then
    Echo
    Echo Type EXIT to return to the Menu when done.
    Echo
    SetVar OLDPROMPT  HPPROMPT
    SetVar HPPROMPT  "EXIT to return to Menu:"
    CI.PUB.SYS
    SetVar HPPROMPT  OLDPROMPT
  ElseIf CHOICE <> 0
    Echo
    If RESPONSE = " " then
      Echo    Your selection of "!WHICHCHOICE" is not a valid response.
    Else
      Echo    Your selection of "!RESPONSE" is not a valid response.
    EndIf
    Echo    Please re-enter your selection.
  EndIf
  If CHOICE <> 0 then
    SetVar CHOICE  Warn
  EndIf
EndWhile
Bye
```

Since the command file has a parameter, the user may indicate which function to perform without the necessity of displaying the menu the first time. If no parameter value is provided, the menu will be displayed, and the user prompted for their selection. Unless the user has chosen to leave the menu, the variable CHOICE is set to the mnemonic WARN at the end of the command file to force the menu to be displayed. NUMERIC(...) is used to determine if the user typed anything other than digits; this will be false if they did. Also note the provision for temporarily exiting to the MPE/iX command interpreter. From here the user may enter MPE/iX commands, but may not run a program unless they have the process handling (PH) capability.

We have a few new predefined variables to talk about, so let us get started. The HPMSGFENCE variable is used to control error messages printed by the CI. If this variable is set to zero, the CI displays all errors and warnings as usual. However, if it is set to one, the CI will only display error messages, and if set to two, all CI messages are suppressed. By setting HPAUTOCONT to TRUE, you can avoid the need to place CONTINUE statements throughout. It is like having a CONTINUE in front of every command. HPMONTH, HPDATE, and HPYEAR provide the month number, day of month, and year respectively. The current hour and minute is provided by HPHOUR and HPMINUTE. HPCIERR contains the last CI error or warning number from your job or session. HPPROMPT is used here to provide a reminder of how they get back to the menu from the CI. You can use virtually any character string for the prompt – including any of the predefined variables!

The inside WHILE loop is an interesting feature. It contains a timed INPUT command that will display the current date and time in front of the prompt.

Because of the timed read and the loop, the time will stay up-to-date. The escape sequence will delete the last line before displaying it again. Also note, if the user does not respond to the command file before the number of minutes specified in HPTIMEOUT has passed, the user will automatically be logged off the system.

Hey system managers; do you have things you want to be performed every time a user logs on? Are there varying activities for certain accounts or users that should always be done? You can provide this functionality with logon UDCs. You say UDCs are a bit cumbersome, and have to be uncatalogued/catalogued every time you want to make a change? Well there is an alternative – sort of. The solution still involves a logon UDC, but it does address the cumbersome issue. I have seen this approach before, but the reference I found when developing this paper appeared in the "Users Forum" column of the September 1982 issue of *Interact* magazine. I have made a few refinements, and present it here for your consideration.

```
SYSLOGON
Option LOGON,NOLIST,NOBREAK
Comment    File:   SYSUDC
Comment    Source: INTERACT 9/92 Page 12
Comment
Comment  System level logon commands.
If finfo("LOGONCMD.PUB.SYS","EXISTS") then
  Xeq LOGONCMD.PUB.SYS
EndIf
Comment  Account level logon commands.
If HPACCOUNT <> "SYS" and finfo("LOGONCMD.PUB","EXISTS") then
  Xeq LOGONCMD.PUB
EndIf
Comment  User level logon commands.
If HPHGROUP <> "PUB" and finfo("LOGONCMD.!HPHGROUP","EXISTS") then
  Xeq LOGONCMD.!HPHGROUP
EndIf
**
```

Command files do not provide a LOGON option, so if we want this functionality, we must use at least one UDC, and the above UDC is it. It does nothing except execute three command files: one each to imitate a system level, account level, and user level logon UDC. Each of the command files then contain all of the programs that need to be run, and settings that need to be made, for each level at logon time. To change what happens at a given level, simply change the appropriate command file and you are done. This can even be done while people are logged on. You will note the UDC confirms the existence of the command file before it tries to execute it, so you are not forced to have a file at every level for every account or user. Also note that what command file(s) actually get executed depends on the logon group and account, since it is based on the logon account and user name.

Here is a sample of what the LOGONCMD.PUB.SYS file might look like.

```
Comment    File: LOGONCMD.PUB.SYS
Comment    Original Source: INTERACT 9/92 Page 12
Setvar HPSYSNAME  "N.G. Gilbert Corporation - Muncie"
If "!HPHGROUP" <> "PUB" then
  Setvar HPPATH  "PUB,PUB.SYS,CMD.SYS,!HPHGROUP"
Else
  Setvar HPPATH  "PUB,PUB.SYS,CMD.SYS"
EndIf
Echo
Echo The following file equations have been set:
Option List
File LP;Dev=LP
File T;Dev=TAPE
```

Nothing fancy here; just a list of commands to be performed for every user that logs onto the system. If you have a logon security system, commands to initiate it would be placed here. You can create a similar command file for each account on your system; each user too! You can effectively keep the user from the CI prompt, thus providing some level of security for free, by having one of these start another command file or a program to place the user into their application. The application must then be followed with a BYE command to log the user off. Do not forget to use the NOBREAK option to disable the BREAK key.

Have you ever been frustrated with yourself when you logged off because you typed EXIT at the CI prompt and did not realize you were running your first level of the CI? Another problem you may have is forgetting to return the console to its rightful place before logging off. Well, here are a couple of solutions to those problems. These will be most effective as system-level UDCs. In fact, they must be set up as UDCs. If you try to use command files for them, they will never be performed because UDCs and MPE/iX commands are performed before command files unless you use the XEQ command.

```
Exit
Option NoList
Comment    File: EXITUDC
Comment    Original Source: INTERACT 3/92 Page 12
Comment  This UDC MUST be defined above the BYE UDC.
ErrClear
If HPINTERACTIVE and HPCIDEPTH = 1
  SetVar ANSWER  "N"
  SetVar HPMSGFENCE  2
  Input ANSWER, "      Do you really want to logoff? (N/Y)", 15
  SetVar HPMSGFENCE  0
  If HPCIERR = -9003 then
    Echo      Timed 15-second read expired.  A "no" answer was assumed.
  ElseIf ups("!ANSWER") = "Y" or ups("!ANSWER") = "YES" then
    Comment  Perform BYE UDC...
    Bye
  EndIf
  DeleteVar ANSWER
Else
  Exit
EndIf
**
```

```
Bye
Option NoList
If HPLDEVIN <> 20 and HPLDEVIN = HPCONSOLE then
  Console 20
EndIf
Bye
**
```

Notice the use of the INPUT command to get a response from the user in the EXIT UDC. HPINTERACTIVE has a value of TRUE if the logon is interactive (i.e. a session). In case you are not aware, you can run the CI from within the CI from within the CI, etc.. To back up to the previous copy of the CI, you use the EXIT command. If EXIT is used at the first level, however, it performs as if you had typed BYE. This UDC gives you a second chance to change your mind before MPE/iX logs you off. The UPS function performs an upshift of the character string given it.

The BYE UDC (which is also referenced in the EXIT UDC) provides the check for the console. By looking at HPLDEVIN and comparing that against HPCONSOLE, which contains the LDEV number of the system console, the UDC is able to determine if it needs to transfer the console back to its normal home before logging you off. Again, please note that these must both be defined as UDCs, and that the definition of the BYE UDC must occur after the EXIT UDC, so that when BYE is referenced in EXIT, the BYE UDC will be performed, and not the BYE command.

Here is a fun little command file (actually it has some size to it). Have you ever wanted a stopwatch functionality on the HP 3000? If yes, this command file is for you! Take some *time* to look.

```
parm action="none"
Comment     File:    SW
Comment     Source:  INTEREX CSLXL
comment*****************************************************************
comment SW.CMD.SYSTEM:  A Stop Watch for MPE/XL Command Interpreter
comment
comment This command file simply reports elapsed time from one point
comment to another.  The "action" parameter is used to control the
comment stop watch functions.
comment
comment       START:    Start the stop watch
comment
comment       REPORT:   Shows the elapsed time since "start"
comment
comment       RESET:    Zeros the stop watch to setup for a new timing
comment                 run
comment
comment Author:    Guy Smith
comment Company:   Circuit City Stores, Inc.
comment Last rev:  07/17/91

.....(continued).....
```

```
.....(continued).....

comment Warning! This command file assumes MPE/XL 2.2 or better.
comment******************************************************************
setvar action ups("!action")
if action="START" then
    if bound(SW_START) then
         comment ********************************************************
         comment Existence of SW_START indicates error.  Let user know
         comment ********************************************************
         echo StopWatch has already been started.  Use RESET first to
         echo zero timer, then START to begin timing
    else
         comment ********************************************************
         comment StopWatch is not running, so start-it-up
         comment ********************************************************
         echo StopWatch has started at !hptimef
         setvar SW_START HPCONNSECS
    endif
elseif action="REPORT" then
    if bound(SW_START) then
         comment ********************************************************
         comment SW_START exists, so it's OK to calculate elapsed time
         comment ********************************************************
         comment First, get current connect seconds before processing
         comment ********************************************************
         setvar SW_STOP  HPCONNSECS
         setvar SW_DELTA SW_STOP-SW_START
         comment ********************************************************
         comment Now calculate elapsed days, hours, minutes, seconds
         comment ********************************************************
         setvar SW_DAYS     SW_DELTA/86400
         setvar SW_HOURS   (SW_DELTA-SW_DAYS*86400)/3600
         setvar SW_MINUTES (SW_DELTA-SW_DAYS*86400-SW_HOURS*3600)/60
         setvar SW_SECONDS (SW_DELTA-SW_DAYS*86400-SW_HOURS*3600-    &
                            SW_MINUTES*60)
         echo StopWatch elapsed time   Day:      !SW_DAYS
         echo                          Hours:    !SW_HOURS
         echo                          Minutes:  !SW_MINUTES
         echo                          Seconds:  !SW_SECONDS
    else
         comment ********************************************************
         comment Tried to STOP without STARTING, so tell 'em
         comment ********************************************************
         echo
         echo StopWatch must be STARTed in order to report an elapsed
         echo time.
    endif
elseif action="RESET" then
    if bound(SW_START) then
       deletevar SW_@
    endif
    echo StopWatch has been reset
else
    comment
    comment User has specified an unknown action
    comment
    echo
    echo StopWatch can't figure out what you want.
    if action="NONE" then
         echo You failed to tell me to take any kind of action.
    else
         echo Action !action is unknown.
    endif
    echo I won't do anything.
endif
```

Guy Smith has done a nice job on this one. I have not found much to change in it. The use of the command file and how it works is fairly straightforward. Take a look at Guy's comments in the command file – they pretty well spell out what is going on. The only new predefined variable is HPCONNSECS which provides the number of elapsed seconds since your log on time.

Here is an example of using the IF command and a variable to do something they were not designed for, but it works, so why not? The following could be included at any point in a job stream where HPDAY would not be equal to 8 (which would be anywhere – unless you have figured out how to get more days in the week than I have!):

```
!If HPDAY = 8 Then
   From this point on (up to an ELSE, ELSEIF, or ENDIF command), you can type
   whatever you wish.  The lines do not even need to start with an exclamation
   point!  The reason this works is that when the condition in an IF command is
   false (which it is in this instance), all command lines are ignored until an
   ELSE, ELSEIF, or ENDIF command is read.  Thus, this provides an easy way to
   include comments without using the COMMENT command.  The "IF" could actually
   be written "If 1 = 2 Then" (or anything that would evaluate to false), but
   then it wouldn't be an example of variable usage!
!EndIf
```

OK operator-type people, how many of you have ever needed to raise the job limit so a job could start running, but then lowered it right after it started? I thought so. Well here is a little command file you might find helpful. I found this in the "Questions & Answers" column of the March 1992 issue of *Interact* magazine.

```
Parm JDELTA = 0
Comment   File:   JOBLIMUP
Comment   Source: INTERACT 3/92 Page 12
Limit ![!HPJOBLIMIT+!JDELTA]
Pause 5
Limit ![!HPJOBLIMIT-!JDELTA]
```

There is nothing real sophisticated here; we simply take the current job limit, add a value to it, wait five seconds, and then set the job limit back to the original value. In case you are curious, yes you can give it a negative value for JDELTA. It temporarily LOWERS the limit and then raises it – I am not sure why you would want to do that, but it does work.

Here is a little command file to help with the CONSOLE command. The "Questions & Answers" column of the March 1992 issue of *Interact* magazine gave me the foundation, but I've elaborated on the original a bit.

```
Parm LDEV=" "
Comment    File: CONS
Comment    Original Source: INTERACT 3/92 Page 12
If ups("!LDEV") = "ME" then
  If HPLDEVIN = HPCONSOLE then
    If HPJOBNAME = ""
      Echo     You already have the console !HPUSER.
    Else
      Echo     You already have the console !HPJOBNAME.
    EndIf
  Else
    Console !HPLDEVIN
  EndIf
Else
  Console !LDEV
EndIf
```

There are three modes of operation for this command file. If you do not provide a parameter, it simply responds with the LDEV number of the location of the console. If you specify an LDEV number, it will attempt to move the console to that device. Lastly, if you provide the word "me", it will attempt to relocate the console to your terminal. This provides a way of getting the console with out needing to know the LDEV number of the terminal you are logged on. To accomplish all of this, it simply works through a series of IF commands.

A simple addition to the system-level logon UDC (or the command file executed from it) makes use of the predefined variable HPDAY easier. By including this functionality at the system level, the following seven variables will always be available for use in IF commands.

```
SetVar Sunday = 1
SetVar Monday = 2
SetVar Tuesday = 3
SetVar Wednesday = 4
SetVar Thursday = 5
SetVar Friday = 6
SetVar Saturday = 7
```

Here is a way to control when people can logon and play games (or what ever it is that you might need to control). This needs to be either included in a logon UDC, or in a command file executed from one.

```
If HPDay = Sunday or HPDay = Saturday or &
   HPHour < 8 or HPHour > 17 or HPHour = 12 Then
  If HPUSERCOUNT < (HPUSERLIMIT / 2) Then
    Echo
    Echo Welcome to the Game Room.
  Else
    Echo
    Echo Sorry, the system is too busy at this time to allow
    Echo you to play games.  Please try again later.
    Bye
  EndIf
Else
  Echo
  Echo Sorry, the Game Room is closed.
  Echo Hours: Saturday and Sunday:  all day
  Echo        Monday-Friday: Before 8 a.m., After 5 p.m.
  Echo                       and Noon to 1 p.m.
  Bye
EndIf
```

If it is Saturday or Sunday, or before 8 a.m., after 5 p.m., or sometime during the noon hour, this will let the user stay logged on – any other time the user will automatically be logged off. Note that if the user gets logged on during an "open" time, they can continue playing forever – there is nothing to force them off when the game room closes. To accomplish this, a check could be added to each of the commands set up to run the individual games. The comparison of HPUSERCOUNT against HPUSERLIMIT simply keeps people out of the games logon if more than half of the available logons have been used. Again, there is nothing to force them off if that becomes true after they have logged on. The first IF command assumes the existence of the variables for the days of the week. If they are not available, the correct numbers will need to be substituted in the IF command.

For more examples, take a look in the INTEREX CSL (primarily in the COMMAND group). There are a number of interesting command files there. I am sure there are some CSL programs that utilize the predefined variables, although I can not point you to any specific examples. Another good source of examples is virtually any job stream from Hewlett-Packard Company. They tend to make extensive use of variables to control the flow of the job stream logic, and many of them are the predefined variables.

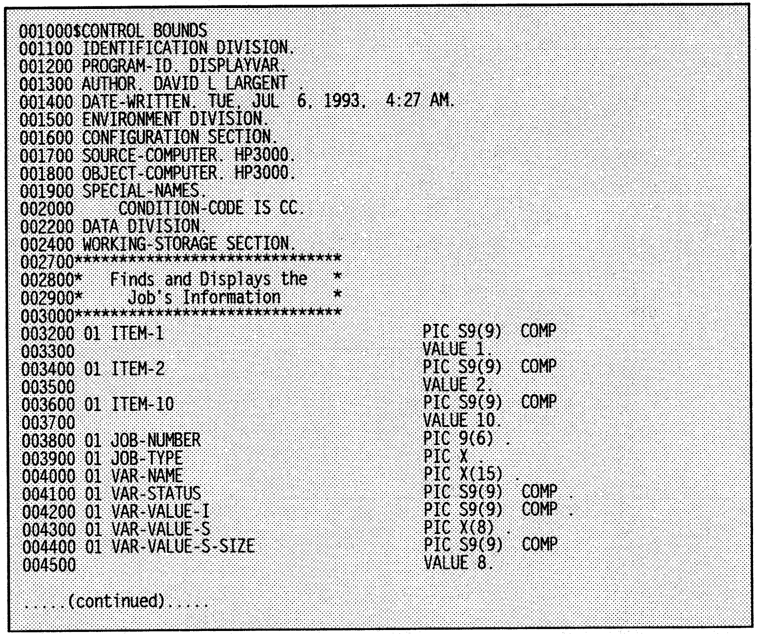
2.4.2 Programmatic examples.

The following is a COBOL II/XL program that illustrates how to use the HPCIPUTVAR intrinsic. As it is presented, it allows the user to provide a value for the CI prompt (instead of the colon, for example), and the CI time out variable.

```
001000$CONTROL BOUNDS
001100 IDENTIFICATION DIVISION.
001200 PROGRAM-ID. PUTVAR.
001300 AUTHOR. DAVID L LARGENT .
001400 DATE-WRITTEN. TUE, JUL  6, 1993,  5:53 AM.
001500 ENVIRONMENT DIVISION.
001600 CONFIGURATION SECTION.
001700 SOURCE-COMPUTER. HP3000.
001800 OBJECT-COMPUTER. HP3000.
001900 SPECIAL-NAMES.
002000     CONDITION-CODE IS CC.
002200 DATA DIVISION.
002400 WORKING-STORAGE SECTION.
002700*****************************
002800*   Sets the HPPrompt and    *
002900*    HPTimeOut Variables     *
003000*****************************
003200 01 ITEM-1                              PIC S9(9)  COMP
003300                                        VALUE 1.
003400 01 ITEM-2                              PIC S9(9)  COMP
003500                                        VALUE 2.
003600 01 ITEM-11                             PIC S9(9)  COMP
003700                                        VALUE 11.
003800 01 TIME-OUT                            PIC 9(3) .
003900 01 VAR-NAME                            PIC X(15) .
004000 01 VAR-STATUS                          PIC S9(9)  COMP .
004100 01 VAR-VALUE-I                         PIC S9(9)  COMP .
004200 01 VAR-VALUE-S                         PIC X(20) .
004300 01 VAR-VALUE-S-SIZE                    PIC S9(9)  COMP
004400                                        VALUE 20.
004600 PROCEDURE DIVISION.
004800 MAIN-LINE-SECTION SECTION.
005000 MAIN-LINE.
005200     DISPLAY "What do you want the prompt set to?".
005300     ACCEPT VAR-VALUE-S.
005400     MOVE "HPPROMPT "                 TO VAR-NAME.
005500     MOVE ZERO                        TO VAR-STATUS.
005600     CALL INTRINSIC "HPCIPUTVAR" USING VAR-NAME VAR-STATUS ITEM-2
005700       VAR-VALUE-S ITEM-11 VAR-VALUE-S-SIZE.
005800     IF VAR-STATUS NOT = ZERO
005900       DISPLAY "Program PUTVAR: " VAR-NAME " not set."
006000     ELSE
006100       DISPLAY "Program PUTVAR: The prompt was set to "
006200         VAR-VALUE-S
006300     .
006400     DISPLAY "What do you want the time out length set"
006500       " to? (3 digits)     ".
006600     ACCEPT TIME-OUT.
006700     MOVE TIME-OUT                    TO VAR-VALUE-I.
006800     MOVE "HPTIMEOUT "                TO VAR-NAME.
006900     MOVE ZEROS                       TO VAR-STATUS.
007000     CALL INTRINSIC "HPCIPUTVAR" USING VAR-NAME VAR-STATUS ITEM-1
007100       VAR-VALUE-I.
007200     IF VAR-STATUS NOT = ZERO
007300       DISPLAY "Program PUTVAR: " VAR-NAME " not set."
007600     ELSE
007700       MOVE VAR-VALUE-I               TO TIME-OUT
007800       DISPLAY "Program PUTVAR: The time out length was set to "
007900         TIME-OUT
008000     .
008100     STOP RUN.
```

I will have to admit this is not a very useful program. It does however, show the basics of what needs to be done to create and/or set a particular variable to a given value. If your variable name is longer than fourteen characters, make sure you increase the length of field VAR-NAME. Also, make sure you have at least one blank or other non-alphanumeric character following your variable name.

Here is another example COBOL II/XL program with the HPCIGETVAR intrinsic, which shows the basics of what needs to be done to retrieve the value of an existing variable.



```
001000$CONTROL BOUNDS
001100 IDENTIFICATION DIVISION.
001200 PROGRAM-ID. DISPLAYVAR.
001300 AUTHOR. DAVID L LARGENT .
001400 DATE-WRITTEN. TUE, JUL  6, 1993,  4:27 AM.
001500 ENVIRONMENT DIVISION.
001600 CONFIGURATION SECTION.
001700 SOURCE-COMPUTER. HP3000.
001800 OBJECT-COMPUTER. HP3000.
001900 SPECIAL-NAMES.
002000     CONDITION-CODE IS CC.
002200 DATA DIVISION.
002400 WORKING-STORAGE SECTION.
002700*****************************
002800*  Finds and Displays the   *
002900*     Job's Information     *
003000*****************************
003200 01 ITEM-1                           PIC S9(9)  COMP
003300                                     VALUE 1.
003400 01 ITEM-2                           PIC S9(9)  COMP
003500                                     VALUE 2.
003600 01 ITEM-10                          PIC S9(9)  COMP
003700                                     VALUE 10.
003800 01 JOB-NUMBER                       PIC 9(6) .
003900 01 JOB-TYPE                         PIC X .
004000 01 VAR-NAME                         PIC X(15) .
004100 01 VAR-STATUS                       PIC S9(9)  COMP .
004200 01 VAR-VALUE-I                      PIC S9(9)  COMP .
004300 01 VAR-VALUE-S                      PIC X(8) .
004400 01 VAR-VALUE-S-SIZE                 PIC S9(9)  COMP
004500                                     VALUE 8.

.....(continued).....
```

```
.....(continued).....

004700 PROCEDURE DIVISION.
004900 MAIN-LINE-SECTION SECTION.
005100 MAIN-LINE.
005300     MOVE "HPJOBNAME "             TO VAR-NAME.
005400     MOVE ZERO                     TO VAR-STATUS.
005500     MOVE SPACES                   TO VAR-VALUE-S.
005600     CALL INTRINSIC "HPCIGETVAR" USING VAR-NAME VAR-STATUS ITEM-2
005700       VAR-VALUE-S ITEM-10 VAR-VALUE-S-SIZE.
005800     IF VAR-STATUS NOT = ZERO
005900       DISPLAY "Program DISPLAYVAR: " VAR-NAME " not found."
006000     ELSE
006100       DISPLAY "Program DISPLAYVAR: The job/session's name is "
006200         VAR-VALUE-S
006300     .
006400     MOVE "HPUSER "                TO VAR-NAME.
006500     MOVE ZERO                     TO VAR-STATUS.
006600     MOVE SPACES                   TO VAR-VALUE-S.
006700     CALL INTRINSIC "HPCIGETVAR" USING VAR-NAME VAR-STATUS ITEM-2
006800       VAR-VALUE-S ITEM-10 VAR-VALUE-S-SIZE.
006900     IF VAR-STATUS NOT = ZERO
007000       DISPLAY "Program DISPLAYVAR: " VAR-NAME " not found."
007100     ELSE
007200       DISPLAY "Program DISPLAYVAR: The user's name is "
007300         VAR-VALUE-S
007400     .
007500     MOVE "HPJOBTYPE "             TO VAR-NAME.
007600     MOVE ZERO                     TO VAR-STATUS.
007700     MOVE SPACES                   TO VAR-VALUE-S.
007800     CALL INTRINSIC "HPCIGETVAR" USING VAR-NAME VAR-STATUS ITEM-2
007900       VAR-VALUE-S ITEM-10 VAR-VALUE-S-SIZE.
008000     IF VAR-STATUS NOT = ZERO
008100       DISPLAY "Program DISPLAYVAR: " VAR-NAME " not found."
008200       MOVE SPACES                 TO JOB-TYPE
008300     ELSE
008400       MOVE VAR-VALUE-S            TO JOB-TYPE
008500     .
008600     MOVE "HPJOBNUM "              TO VAR-NAME.
008700     MOVE ZEROS                    TO VAR-STATUS
008800                                      VAR-VALUE-I.
008900     CALL INTRINSIC "HPCIGETVAR" USING VAR-NAME VAR-STATUS ITEM-1
009000       VAR-VALUE-I.
009100     IF VAR-STATUS NOT = ZERO
009200       DISPLAY "Program DISPLAYVAR: " VAR-NAME " not found."
009300       MOVE ZERO                   TO JOB-NUMBER
009400     ELSE
009500       MOVE VAR-VALUE-I            TO JOB-NUMBER
009600     .
009700     DISPLAY "Program DISPLAYVAR: The job/session # is " JOB-TYPE
009800       JOB-NUMBER.
009900     STOP RUN.
```

Same comments as last time; make sure you set up VAR-NAME large enough to hold your variable name and make sure you end it with at least one blank or other nonalphanumeric character.

3 *Closing thoughts – "For a good time... check out HPSusan!"*

We have looked (stared?!) at HPSUSAN and her harem, the MPE/iX predefined variables. Numerous examples have been explored to see how they work and how they can be used. We have found that they will tell us virtually anything we might want to know – except how old they are. They are a very powerful feature of the HP 3000 and MPE/iX.

So, is it worth the effort of learning something new? My answer is a resounding YES! Use of the predefined variables must be carefully planned and monitored to reap the greatest benefit, but, oh, what a benefit it is: Increased operator, programmer, and user productivity and a computer system that is easier to use overall.

HP is a trademark of Hewlett-Packard Company.

Acknowledgements

A special thanks go to the following people who assisted in creating this paper in some way – all of their efforts (and tolerance!) are greatly appreciated:

Dan Hinds *Jonathan Largent* *Julia Largent* *Lois Largent* *Jamie Lunsford*

Bibliography

Cressler, Scott; Vance, Jeff; and Elmer, Steve
"Advanced CI Programming: The 2.1 Story", *Interact*, February, 1991, page 83ff.

Dunlop, John
"CI Programming in MPE XL/iX or What the Manuals Don't Say", *Interact*, November, 1992, page 78ff.

Hewlett-Packard Company
"Command I/O Redirection", *Communicator 3000/XL*, Volume 6, Issue 1, May, 1991, page 9-10ff.

Hewlett-Packard Company
Command Interpreter Access and Variables Programmer's Guide, Second Edition, 1990, All Chapters.

Hewlett-Packard Company
Command Interpreter Programming Quick Reference Card, December, 1990.

Hewlett-Packard Company
Getting System Information Programmer's Guide, First Edition, Update 1, 1988, Chapter 3.

Hewlett-Packard Company
MPE V to MPE XL: Getting Started, Third Edition, 1989, Chapter 5 & 7.

Hewlett-Packard Company
MPE XL Intrinsics Reference Manual, Third Edition, 1990, Chapters 2 & 4.

Hewlett-Packard Company
MPE/iX Commands Reference Manual Volume I, Fifth Edition, 1992, Chapter 2.

Hewlett-Packard Company
MPE/iX Commands Reference Manual Volume II, First Edition, 1992, Chapter 2 and Appendix A.

Hewlett-Packard Company Mountain View Response Center
Calls to the Mountain View HP Response Center during June, 1993.

Hewlett-Packard Company North American Response Centers
HP3000 Application Note #21: COBOLII and MPE Intrinsics, January 15, 1987.

Hewlett-Packard Company North American Response Centers
HP3000 Application Note #94: RPG/XL Intrinsic Interface, January 15, 1992.

INTEREX
INTEREX Contributed Software Libraries (1990-1991).

Largent, David L.
"Function Key Labelling", *Interact*, February, 1988, page 15.

Largent, David L.
"JCWs: An Introduction", *Interact*, April, 1991, page 36ff.

Manise, Walter
"Questions & Answers", *Interact*, March, 1992, page 12ff.

Parker, Michael J. and Wilson, Lynn
"Labelling F-keys", *Interact*, November, 1987, page 22ff.

Pratt, Bruce
"UDCs vs. Command Files", *Interact*, September, 1992, page 10ff.

PAPER NUMBER: 5022

TITLE: Securing a DOE Installation

PRESENTER: Geroge Malcolm
Brookhaven National Laboratory
Building 459
Upton, NY 11973
516-282-7654

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Paper #5023

Performance Issues With Large Disks

Sam Yamakoshi
Hewlett-Packard
19111 Pruneridge Ave.
Cupertino, CA 95014
(408) 447-5586

INTRODUCTION

Following the recent introduction of the disk arrays, a number of questions and performance issues have risen. Having worked with Hewlett-Packard Disk Memory Division in Boise, Idaho and a number of customer sites which experienced some difficulty, I will highlight some of performance issues along with the outcome of working with these customer sites (case studies). These issues and guidelines should help with future disk products, and the positioning of these larger mass storage devices with HP customers.

The disk array drives are the products: HP C2252 ("2-way", 2.72 Gbyte) and the HP C2254 ("4-way", 5.44 Gbyte). Both of these products have the option to add an additional mechanism which provides high availability in the event one of the internal mechanisms fail. In the tests conducted in this report, all drives had the additional parity mechanism (there is no perceived performance degradation in doing so).

Our group, System Technology Division, I/O and System Modeling has been working together with DMD (Disk Memory Division) for a number of years and have characterized the performance of other peripherals as they were introduced.

OBJECTIVES

Identify the key performance issues related to the disk array drives (on NIO and CIO), some of the difficulties to be aware of, and guidelines on positioning the drives with different computer system and configurations.

SOFTWARE ENVIRONMENT

The version of MPE/XL described in this document is Release 3. The case studies described later each had a version of Release 3 with disk array support.

HARDWARE ENVIRONMENT

-NIO versus CIO

In a previous report that I had written, I discussed the performance of disk arrays on a Series 980, a CIO Input/Output subsystem.

Here is a simple illustration of one CIO hardware configuration. The maximum bandwidth for each of the hardware components is listed in parentheses. These should not be construed as actual limits. The actual bandwidth achieved is, in most cases, less than this maximum megabytes per second.

The peripherals (disks, tapes, printers, etc.) are connected to either HP-IB, SCSI, or fiberlink device adapters. The bandwidth of the card becomes a factor when combined transfer rate of the devices exceeds (limited by) that of the card.

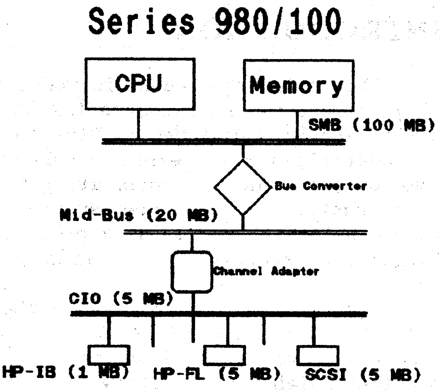

Series 980/100
CPU
Memory
SMB (100 MB)
Bus Converter
Mid-Bus (20 MB)
Channel Adapter
CIO (5 MB)
HP-IB (1 MB)
HP-FL (5 MB)
SCSI (5 MB)

Figure 1 - CIO Configuration

In contrast, this illustration of an NIO system shows a larger bandwidth at that level of the hardware IO subsystem.

It is theoretically possible then to reach a bottleneck on the CIO card with a large number of peripherals being configured and accessed. The illustrated bandwidths are not what is currently attainable today. We have measured the hardware IO subsystem at each of the levels under maximum burst conditions (with different block sizes) and achieve transfer rates less than what is indicated.

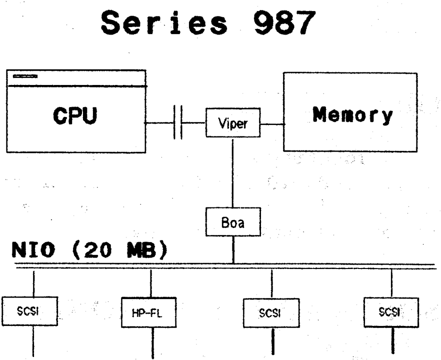

Series 987
CPU
Viper
Memory
Boa
NIO (20 MB)
SCSI
HP-FL
SCSI
SCSI

Figure 2 - NIO Configuration

HARDWARE ENVIRONMENT

- NIO versus CIO - continued

In many instances of hardware IO bottleneck, it appears that the device adapter card is the limiting factor before the CIO. For instance, it would take a disk IO rate of 400 to 500 per second at a blocksize of 10 Kbytes to get a rate of 5 Mbytes per second (78 per second at 64 Kbytes) across several device adapter cards. With the device adapter cards (fiberlink, HP-IB, and current SCSI), as the blocksize gets smaller, the maximum IOs per second is less because of overhead required at different levels. Applications must spread the disk access over many more drives before reaching the maximum capacity of CIO (even with multi-processor Series 980s). For future systems, however, NIO removes a possible bottleneck in the IO subsystem.

Another factor in characterizing the performance of disk arrays on an NIO system versus CIO is that there is another layer of software that must be executed when accessing the CIO disk drive.

The illustration above shows the layers of software required to access the fiberlink disk drives on an NIO versus a CIO hardware IO subsystem. The number of instructions from the external interrupt to get to the disk drive is 5037 for NIO as compared to 10974 instructions on a CIO system. This does not include the File System which precedes the interrupt.

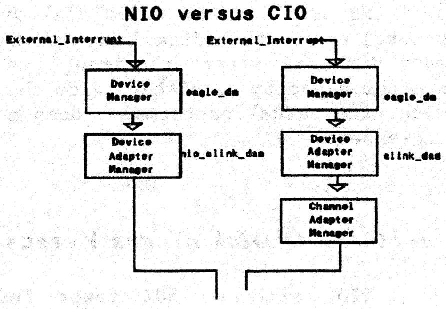

NIO versus CIO
External_Interrupt
External_Interrupt
Device Manager
eagle_dm
Device Manager
eagle_dm
Device Adapter Manager
nio_alink_dam
Device Adapter Manager
alink_dam
Channel Adapter Manager

Figure 3 - NIO v CIO Software Path

The disk exerciser tests that I ran on an NIO system (Series 977) showed no difference (no measurable improvement) over the tests run on a CIO machine (Series 980). The exerciser program I wrote, calls SENDMMIO which will initiate the disk driver (eagle_dm), bypassing the File System and main memory manager caching (generating all physical disk IOs). The number of additional instructions executed on these systems did not change the transfer rates.

DISK PERFORMANCE TESTS

- *Disk Mechanisms*

Before comparing the HP C2204 (1.3 Gbyte) disk drives and the HP C2252 and HP C2254 disk arrays, here is a summary chart that I included in a previous report that I distributed:

DESCRIPTION	Eagle 1	Coyote 2	Coyote 3
Capacity (Mbytes)	571	670	1300
Track Size (Kbytes)	30.75	28.25	30.27
Full Latency (ms)	16.67	14.99	14.99
Track Speed (Mbyte/sec)	1.845	1.885	2.303
Random (1/3) Seek (ms)	20.5	17.0	13.5

The Eagle mechanism is used in HP 7936/37 disk drives, the Coyote 2 mechanism is used in the HP C2203 (HP-IB) and HP C2204 (HP-FL), and the Coyote 3 mechanism is used in the HP C2252 and HP C2254 drives.

HP C2252 and HP C2254 disk drives are (at the time of this report) the fastest disk drives that HP offers. The question is how many disk drives are required by a customer site. Although the storage capacity is either twice that of the HP C2204 or 4 times as much, the actual performance does not correspond to these storage increases.

- *NIO/CIO Disk Arrays Versus HP C2204 - Raw Disk IO*

NIO, although 50% fewer instructions from the external interrupt to starting the IO on the disk drive, showed no measurable gain in performance (difference was less than 5%). Therefore, all discussions of disk arrays in this report will not make a distinction between tests for NIO versus CIO.

DISK PERFORMANCE TESTS

- NIO/CIO Disk Arrays Versus HP C2204 - Raw Disk IO continued

The graphs below show the performance of the HP C2254 disk array versus the HP C2204.

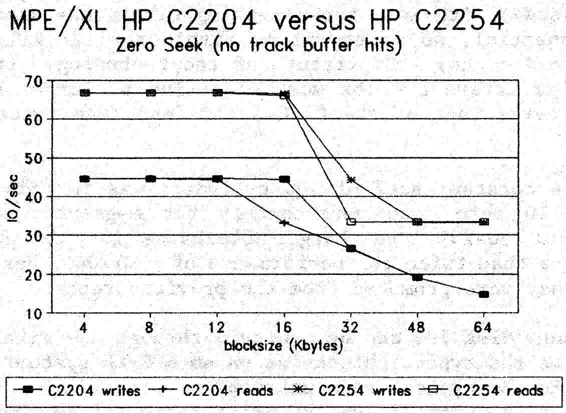


Figure 4 - C2204 v C2254 (IO rate)

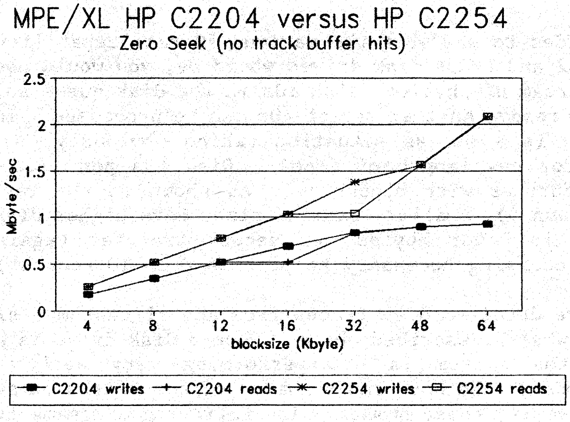


Figure 5 - C2204 v C2254 (Mbyte/sec)

DISK PERFORMANCE TESTS

-NIO/CIO Arrays Versus HPC2204 - Raw Disk IO cont'd

Shown in terms of disk IOs per second (figure 4) and megabytes per second (figure 5) by multiplying disk IOs by blocksize (disregard the steps that occurs at 16 kilobyte blocksize and 32, track switching and other functions took place at these sizes, the curves should be more smooth). These are raw disk IO (bypassing the MPE file system) with zero seek (seeking within the same cylinder but not sequential, so as to not be within the 128 kilobyte disk mechanism read cache). By virtute of the technology, it is shown that the disk arrays are the most effective at larger blocksizes (more than twice that of the C2204) and less than twice at lower blocksizes.

When a constant seek of 300 cylinders was introduced in the tests, the IO rate (and consequently the megabyte per second) dropped about 20-25%, but larger blocksize for the disk array remained more than twice the performance of a C2204. Here are some questions that were prompted from the previous report:

How many disk IOs can be achieved through the File System?
What is the typical blocksize on an MPE/XL system?
What is the typical seek pattern?
What is the typical read to write ratio (since there is a disk mechanism read cache of 128 kilobytes)?
How many reads are processed in this disk controller cache?

-NIO/CIO Disk Arrays Versus HPC2204 - Actual Disk IO

In order to see what the maximum IO rate capability of an HP C2204, C2252 and C2254 disk drives would be, you would need to keep a steady stream of physical disk IOs in the disk queue so that when the disk is ready, another set of IOs can be processed. In customer sites, this is a stress situation (which obviously could not be tolerated for any length of time). Disk IOs per second must be qualified further with blocksize. As shown in the raw disk IOs (figures 4 and 5), smaller transfer sizes have higher disk IO rates but result in lower Mbytes per second transfer (again, due to overhead processing necessary to process each IO request).

I have detailed disk traces from actual customer sites which illustrate what I described above. When a disk drive is at maximum capacity, the system is not performing very well (that's an understatement). We have run a number of benchmarks on stand alone systems to verify these number. The following diagrams illustrates this effect.

DISK PERFORMANCE TESTS

- NIO/CIO Disk Arrays Versus HP C2204 - Actual Disk IO continued

This diagram is an actual trace from a disk drive on an HP 3000 system running MPE/XL release 3.0. The drive is a fiberlink HP C2204 with a disk IO rate of over 22 per second. The average blocksizes are 5.6 Kbytes for reads, 7.2 for writes with a very high read to write ratio. Although the average over time is 22 per second, the requests are coming in at over 40 per second. Over 30 are processed and the requesting processes must wait for some completion before continuing on (this is why it looks somewhat like a cardiac diagram).

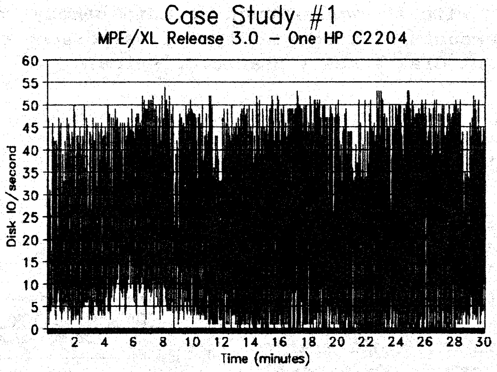


Figure 6 - HP C2204 Disk Trace

So at a blocksize of around 6 Kbytes, the HP C2204 can process around 20 to 22 disk IOs per second. With a larger blocksize, fewer disk IOs per second can be handled. For the HP C2252 and HP C2254, this maximum is around 40 per second. However, when the disk IO rate for a disk array is just below 40 per second with a blocksize around 6 Kbytes, there is a performance problem. Although the drive can process more than 40 per second, there is a great deal of pause and overhead associated with this workload. If you view the activity (similar to figure 6) for a normal running system, you will see periodic bursts of disk requests (sometimes more than 100 IO requests), but as long as it is not sustained for any length of time, you may not even experience the momentary pause.

MPE/XL DISK IO CHARACTERISTICS

-DATA SOURCE

The following data is a summary of 20 disk traces from 8 customer sites. A majority of the traces are systems running MPE/XL release 3.0. Although this may be a small set to be drawing any conclusive information, the data covers the spectrum of small systems with a small amount of main memory to large systems with large amounts of memory. There is also some consistency among the different disk summary characteristics.

-BLOCKSIZE DISTRIBUTION

Illustrated above is an average of 6 of the sites where data was collected. There did not appear to be any significant difference (in the individual distributions) between sites which had large memory and large processors versus systems with small memory. With larger amounts of memory, the average read (due to prefetching) is a bit larger, but not significantly. The average read size was around 8 kilobytes (2 pages) and the average write was around 12 kilobytes (3 pages).

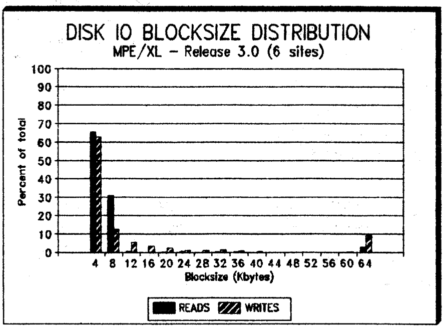


Figure 7 - Disk IO Blocksize Distr

This illustrates that MPE/XL is currently at the smaller blocksize of the charts which graph the performance of the disk arrays. During backup using TurboSTORE, however, the reads from disk are the 60 to 64 kilobyte reads from disk.

The average read to write ratio was between 1.77:1 to 4:1 (reads:writes). However, depending upon what production is going on, we have seen read to write of 1:1 during batch updates and up to 10:1 during daytime on-line transaction processing (just session accessing data in a database with no updates).

MPE/XL DISK IO CHARACTERISTICS

-SEEK DISTRIBUTION

In order to understand the seek distribution for an MPE/XL system, it may be helpful to have an idea of the composition of the disk IO. Without going into a great deal of detail (information for another report), the disk IO summary for these sites is:

DESCRIPTION	Aver %	High %	Low %
NM & CM library ref	6.1	11.4	4.1
File Label Mgmt	2.8	6.7	.4
Loader	3.9	9.0	.5
Process Mgmt	2.8	7.5	1.2
TurboIMAGE	63.9	80.4	45.4
Trans Mgmt Log File	3.4	3.2	3.9
Others	17.1		

Listed are the major components (highest percentage) after breaking down the disk accesses by operating system processes. This is a breakdown of 6 of the customer sites that I have. I selected these because they had the highest utilization of their disk drives and were installing or going to install disk arrays. The disk IO rates ranged from 36 to 185 per second. The processors ranged from Series 949 to 980, and memory sizes were from 128 to 324 megabytes. Obviously, these sites were predominantly TurboIMAGE database users.

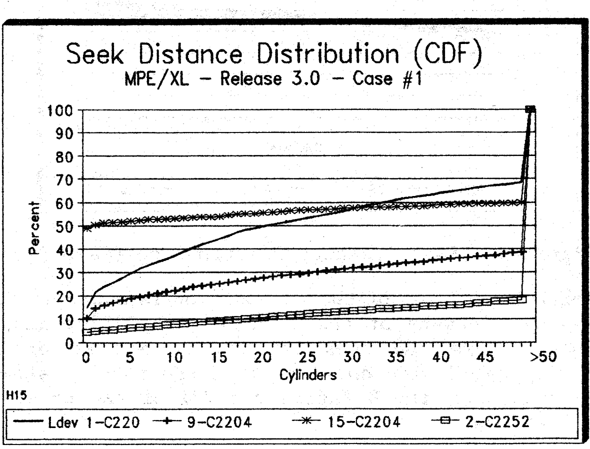


Figure 8 - Seek Distance Distribution (Case #1)

MPE/XL DISK IO CHARACTERISTICS

-SEEK DISTRIBUTION continued

The illustration on the previous page is not the average over many sites, this is an example of one site. The size of the database, the number of master to detail filesets and the patterns of access by the applications, the number of databases accessed simultaneously, the number of different applications running concurrently will all impact the physical disk IO seeking patterns (maybe part of another report?). In figure 8, I have shown logical device 1 separately. On this particular system, 15% of the reads and writes occurred within the same cylinder and around 70% occur within 50 cylinders (in either direction). The combined seek distances of 15 logical devices (C2204s) had 50% of the seeks within a cylinder from the previous access. These disk access are made up of a majority of non-database applications. The 9-C2204s and 2-C2252s demonstrate a predominantly before and after illustration of putting most of the databases onto two C2252 ("2-way") disk arrays. In this case, reducing the number of spindles and putting the filesets onto a larger disk drive caused longer seeks to occur.

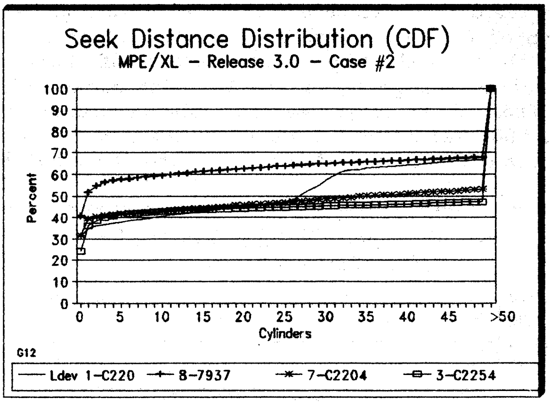


Figure 9 - Seek Distance Distribution (Case #2)

Another illustration of that transition is shown here. Figure 9 is a direct replacement of the 8 Eagles (HP 7937s) and 7-C2204s with 3-C2254s. Here the seek pattern is not as dramatically pronounced. Logical device one has 7% of the total system disk IO rate of 88 per second, the 8 Eagles had 27% of the total, and 66% went to the 7-C2204s.

MPE/XL DISK IO CHARACTERISTICS

-*SEEK DISTRIBUTION continued*

Further analysis showed that none of the 6 sites that had disk arrays showed a read hit rate in the 128 kilobyte controller cache area greater than 3% (due to the random access nature of TurboIMAGE stored on the larger mass storage devices, for these particular applications).

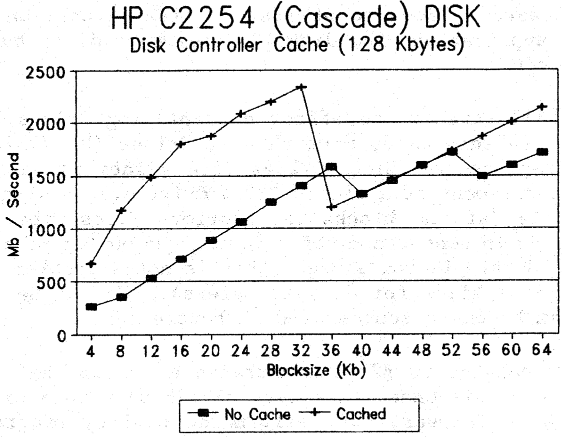


Figure 10 - Disk Mechanism Cache

Figure 10 illustrates the effectiveness of the disk mechanism cache (128 kilobytes) on sequential reads using the raw disk IO generating program. For the single HP C2254, the total cache for reads is 512 Kbytes (4 mechanisms times 128 Kbytes, the fifth mechanism is parity). Below 32 Kbytes, the drive can transfer information at a much higher rate. At larger blocksizes, a larger cache is necessary. However, because of the random nature of TurboIMAGE, only an insignificant amount of the reads are benefitted.

During disk backup, TurboSTORE generates sequential reads but generally at a blocksize of 64 kilobytes.

CASE STUDIES OF DISK ARRAY INSTALLATIONS

-INTRODUCTION

In the process of working with a few of the customer sites that have experienced some difficulties or concerns, I have put together 3 case studies which demonstrate situation which should be avoided.

Case#1 illustrates primarily an allocation problem where disk arrays were added to a system. To isolate the databases targeted for the disk array, a private volume set consisting of 2-C2204s and a C2254 was created. Because of restore file allocation and the nature of the application, the disk IO rates were distributed in a inopportunistic way.

Case #2 illustrates the effect of replacing drives based on storage only. In this case, more than 12 drives (HP 7937s and HP C2204s) were replaced by three C2254s (5.4 gigabytes each). The performance improvement of a single C2254 drive over a single C2204 of around double (at the blocksizes previously described) cannot overcome the loss in concurrency of reducing the number of spindles. When the disk IO rate is low enough, this is not a problem (and may improve the access times for disk transfers). For higher disk IO rates, this can become a source of disk bottleneck.

Case #3 (related to #2) domonstrates a system that replaced 3 disk drives (in this case, they were HP 7937s) with a disk array and the system got slower, but performance utility programs show only 27 IOs per second for the C2254.

-CASE STUDY #1

This particular customer has a Series 949 which has 128 megabytes of memory and no space for additional cards (25+ disk drives). Sometimes the obvious solution of upgrading the processor is not an option to the customer. System was running at 170+ disk IOs per second during production update. Disk arrays were added, and production update time significantly increased. Running several performance utilities showed the C2254 at 16-20 IOs per second. I didn't investigate the uiltities being used (and verify the correct version), but one utility indicated that the disk array was 100% utilized.

I received a disk trace of the system during this production cycle and created the illustrations on the following page. Of the 25+ disk drives, during this production run, only 3 of the disk drives were being heavily accessed.

- CASE STUDY #1 continued

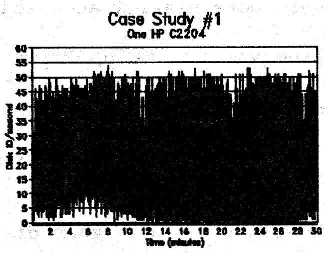


Figure 11 - Case #1: Ldev 41

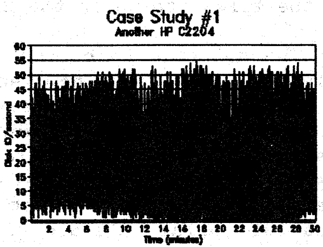


Figure 12 - Case #1: Ldev 42

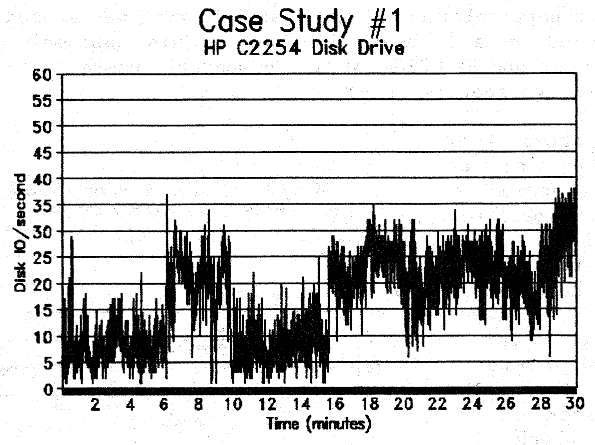


Figure 13 - Case #1 - C2254

The figures above show the two HP C2204s at 20-22 IOs per second, and the HP C2254 at 16-20 IOs per second. One explanation of the erratic ups and downs for the HP C2204s is that the disk IOs come in at 40-50 per seconds and the process is blocked until completion. Compare figures 11 and 12 to figure 13.

CASE STUDIES OF DISK ARRAY INSTALLATIONS

-CASE STUDY #1 continued

These 3 drives were created as a Private Volume set so that restoring the files could control the location of the databases. Files were properly allocated by the size of the device, but looking more closely at the disk addresses in the disk trace, it turned out that the files stored on the HP C2204s were more heavily accessed.

The solution to correct this problem was to set up Private Volume sets made up of homogeneous units. Replacing the HP C2254 with several HP C2252s and removing the HP C2204s from the volume set, restore the databases more evenly across the drives. If the files got restored in a similar fashion where the heavily accessed parts of the database resided on a single disk, we may have had to add another member to the set to redistribute the extents. As it turned out, using multiple "2-way" disk arrays (with higher disk performance than the HP C2204) relieved the stress situation.

-CASE STUDY #2

In this particular case, the system IO rate was around 90 disk IOs per second on a high-end HP 3000. This customer replaced several HP 7937s and HP C2204s with a comparable number of HP C2254s (based on storage requirements).

Figure 14 shows the disk IO rate comparisons between 8 HP C2204 fiberlink disk drives as compared to 2-HP C2254s and then 4 disk arrays. These test results also appeared in the previous report. Each IO required a seek of 300 cylinders, which is not really random (additional results with seek distance of zero was included in the prior report).

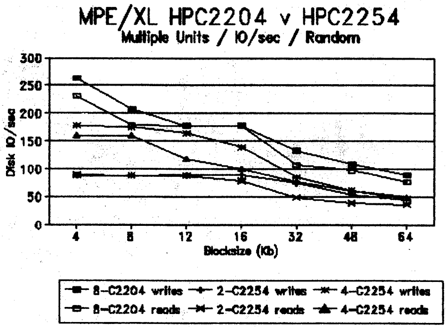


Figure 14 - Multiple Drives (IO rate)

-CASE STUDY #2 continued

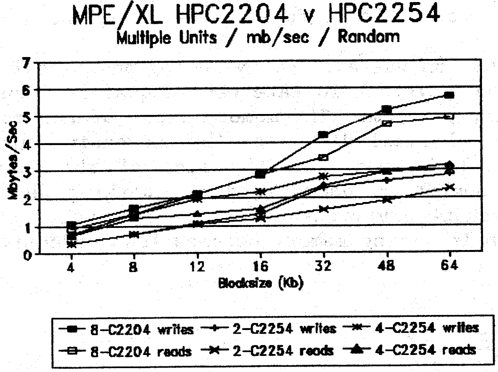


Figure 15 - Multiple Drives (Mbytes/sec)

Here is the same illustration in terms of megabytes transferred per second (instead of disk IOs per second). One qualification is that the 8-HP C2204s were configured on 2 fiberlink device adapter cards, and the 2 and 4-HP C2254s were configured on a single card. These tests were run using a disk exerciser program that I wrote to bypass the memory caching and File System (generate physical disk reads and writes). If the HP C2254s (and possibly the HP C2252s as well) were configured on 2 fiberlink device adapter cards, the total transfer rate would not have leveled (at the higher blocksizes) as they do in figure 15.

However, this figure illustrates that when the disk drives are near or at capacity, the performance ratio can not match the capacity ratio. When storage is required but the access rate is less than half of capacity (both the drives being replaced and the access rate for the disk arrays), a performance improvement can be achieved since the disk array has a faster access time and transfer rate.

For case study #2, if you have a system IO rate of 90 per second on 15 disk drives, the system is performing well as long as the HP C2204s are not much over 12-15 IOs per second (here's where there is a higher probability of disk queues forming). The average disk rate is 6 requests per second (90 divided by 15 drives). When you replace them with 3 disk arrays (maximum 40 IOs per second at 8-10 kilobyte blocksize), you should

CASE STUDIES OF DISK ARRAY INSTALLATIONS

-CASE STUDY #2 continued

not expect any performance gain. In fact, one should set the expectation of a possible performance degradation.

This situation was avoided by adding additional main memory. In doing so, the physical IO rate was adequately reduced to avoid this concurrency problem. The memory issue of how much memory to add to drop the physical disk rate (by increasing the main memory disk cache area) depends upon the size of the databases being accessed, the number of users (and the working set size of the users), how much memory pressure exists currently, etc. At some point, the gain in adding memory becomes less effective.

-CASE STUDY #3

This case involved a site that had a system that was performing well until an HP C2254 disk array was installed (replacing 3 drives). When the customer viewed the disk activity using a number of performance utility programs, the disk drive was at 25 to 27 disk IOs per second. The read to write ratio was about 3:1 with reads at a size of 6.4 kilobytes and writes at 12-16 kilobytes.

	#Disk Arrays/HPFL		#IO/sec/Disk Array	
Blocksize	C2252	C2254	C2252	C2254
4 Kbyte	8	8	22	25
8 Kbyte	8	8	21	24
16 Kbyte	6	6	19	22
64 Kbyte	2	2	13	14

This table appeared in the prior report. The recommendation for an HP C2254 disk array at an 8 kilobyte blocksize is around 24 disk IOs per second. Beyond this, there is a higher probability that disk queues (disk request that currently cannot be satisfied and must wait for processing) are developing.

A disk IO rate of 25-27 per second is likely to have some requests waiting (in the disk queue). The 3 disk drives that were replaced may have had disk IO rates below the recommended limit (i.e., maybe 7-9 per disk drive). Further discussions with support

- CASE STUDY #3 continued

engineers indicated that performance loss was noticed in batch jobs that took longer (wall clock time) to complete. This is due to the disk array having disk queues which did not exist with the 3 replaced disk drives. The sessions during the day were probably getting better response time, but is not noticed by applications (probably in block mode). From the disk traces, it showed that the requests were coming in much higher than the disk drive could process (and the peaks were steady).

25-27 disk IOs per second is not an unbearable situation. Comparing the batch jobs to the way they were running previously, however, may not be desirable.

To avoid this situation, a good guideline is to add the IO rates for the disk drives that are going to be replaced. If the total exceeds the 24-25 recommended disk rate, it may require two HP C2252s as a replacement rather than a single HP C2254.

Additional analysis of the disk traces showed a higher (than the average) number of requests from process management (refer to the table in MPE/XL Disk IO Characteristics, Seek Distance). As it was later learned, the system was configured with limited main memory. Additional memory would reduce the physical disk rate below this range where disk queue were developing.

SUMMARY

The access times for the disk arrays make these devices the fastest disk drives currently available. The parity mechanism provides high availability and is very impressive in how it continues operating when a mechanism fails. The customer expectation must be correctly set if the installed disk arrays are driven to capacity. Below the 24-25 IOs per second the drives are more efficient.

When replacing older disk technologies with the newer large capacity disk drives, look at the current disk IO rates for the disk drives which are going to be removed and see if it exceeds the recommended limit for the incoming larger capacity drives. If the rate is higher, this may require multiple HP C2252s instead of a single HP C2254. If main memory increase is a possibility, this may be enough to allow the larger disk array to be installed. However, one must be aware that adding memory at some point results in diminishing returns.

Using Software
To Automate
Technical Support

Paper #5024

by Ken B. Robertson

Robelle Consulting Ltd.
15399-102A Avenue Unit 201
Surrey, B.C. Canada V3R 7K1
Phone: (604) 582-1700 Fax: (604) 582-1799
E-Mail: ken_robertson%robelle@mci.com

Introduction

« Rinnnnng » *Pause.* « Rinnnnng » *Pause.* « Click »

"Hello, you've reached the Happy Software Technical Support Line. We offer the intelligent telephone support system that no other software company has. Please press one if you have a problem, two if you'd like to connect to our sales phone system."

« boop »

"Thank you for pressing one. If you have an installation problem, please press one. If you have a file-save problem, please press two. If you are having a problem printing your file, please press three.

« fwah-boop »

"Thank you for pressing three. If you have a problem with printing your files to an attached printer, press two. If you have a problem printing your files to an attached LaserJet printer, press three. If you have a problem printing your files to a spooler, please press four."

« bip »

"Printing to an attached LaserJet is a known bug. We are preparing to send the patch file right now. Please wait a moment."

Pause. Pause. Pause.

"Thank you for waiting. Please turn your modem data/voice switch to data to receive the patch file update. Thank you for calling the Happy Software Technical Support Line.

« screeech squawk screeeech screee -- » NO CARRIER

The above scenario doesn't quite represent the automated technical support of the future. And for the next few years, at least, I see the need for human interaction. The current voice-mail systems have inadequate sophistication to handle most problems, and at best should be used only as an answering service.

Technical support today comes in a variety of flavors. We usually hear the term associated with the technical help that a software or hardware company provides its customers, but it can take the form of a centralized "help-desk" in a medium to large company. Such a help-desk would provide support for the company's internal hardware and software. Similarly, agencies exist whose sole purpose is to provide help-desk support for small companies who cannot afford to staff an internal help-desk.

No matter which situation, providing technical support involves the same activities: handling user calls, managing records and resources, and maintaining good communication between departments. In each of these areas, we will see how automation can help any company provide technical solutions with maximum efficiency.

Technical Support Goals

In automating technical support, there are four main goals:

☐ Provide a quick and inexpensive solution for the client

☐ Track the problem call

☐ Create a dialog between technical staff and programmers

☐ Log the problem and the solution

It doesn't matter how the problem gets to the help-desk. It can arrive by fax, by telephone, through electronic mail, or be scrawled on the back of a napkin using ketchup, and handed to a techie in the company cafeteria. All of the above points still hold true for technical support.

Provide a Solution

When a customer or user has a problem, they become irritated. If they call technical support (or tech support, as it is commonly known), the problem should receive immediate attention. A prompt solution will give the caller some perceived value, and they will be happy again, perhaps happier than before. After all, *they* found a bug for you.

Providing a solution quickly will be more cost-effective for you and for the customer. Problems that require many follow-up calls use up a lot of expensive techie time, and may ultimately cause frustration for the caller.

Track the Call

If finding the solution to the problem requires more than a few minutes on hold, the customer will have to be called back. *This call will have to be tracked.* Rarely can a busy help-desk techie hang up the telephone and then devote an entire afternoon to solving that person's problem - it is highly likely that another call will come in, demanding attention. If you don't track the first call, you will lose it - especially on those days when tech support is neck-deep in calls.

Create a Dialog

There will often be trouble-calls that require some communication between tech support and the programmers or engineers. For simplicity's sake, let's limit our discussion to software.

Creating an open dialog between tech support and the programmers is important for a number of reasons. Firstly, the techies often have a better grasp of the types of problems that occur, giving valuable input to the programs and analysts. Secondly, the programmers are made aware that there are problems in their code. If they never see the "work around" that a techie suggests to the client, the errant code will never be changed. Finally, the programmers may come up with the solution to the problem more quickly than the techies.

You don't want to be bothering the programmers with every little problem - they would be very annoyed if their deep-thought programming mode were to be interrupted with a telephone call or a yell over a cubicle baffle. Instead, you should use a form of electronic mail to send them messages.

Log the Problem and Solution

The biggest reason to log problems and their solutions is that it will you save time in the future. When clients call with similar problems, tech support can simply refer to the trouble logs and rapidly provide expert solutions.

With these logs as resources, you can considerably reduce the training time for techies, as well. New techies can't possibly understand all of the problems that have occurred in the past few months of a product, let alone the past few years. However, they can study the trouble logs in their spare moments, and benefit from all the valuable past experiences.

A Help-Desk Scenario

In the next few pages, I'd like to describe some typical help-desk procedures of a mythical company, and the pros and cons of those approaches to support. Perhaps you'll see some similar approaches to help-desk support in your company. We'll look at three types of help-desks: Operations, Tech Support, and Customer Support.

Operations

The Xyzzy Company has an operations help-desk, manned by operators from seven in the morning until eight o'clock at night. These same operators are also responsible for responding to tape requests, printer maintenance, and batch job scheduling, as well as other computer operational duties.

The typical trouble calls they receive are from users who can't log on, whose terminals are locked up, users who are missing reports, or who have severe equipment problems.

A large procedures manual is maintained by the operators, with pencilled notes ready for the next printing update. Some of the calls are handled "on-the-fly", while some take longer to handle. In these cases, a one-line entry is logged to a trouble sheet, and a more detailed description written to a log book.

This is not really a bad system. Having a procedures manual is a must - no one person can remember all of the operational procedures required for every system running on the computers, from coldstarts to monthly batch runs. Logging the trouble call is a good way to provide continuity from shift to shift of operations, as well as to indicate whether calls have been handled or not.

There are some points to consider. In the Xyzzy procedures manual, the operators pencil-in changes. As printed revisions of internal manuals tend to be infrequent, procedures may be lost to the most minor of accidents. "I spilled my coffee on it" is hardly a good excuse for destroying the latest changes to the system configuration charts. The manual should be updated on the computer, and revision pages printed and inserted. Of course, backup tapes should be stored off-site to allow for minor and major disaster recovery.

The paper logging of calls at Xyzzy doesn't help operations adequately in a number of areas: tracking cyclic problems, sustaining the needed dialog between the analysts and programmers who maintain and develop the internal software, and helping operations and system management be pro-active.

I worked for one company who had a very strange cyclic problem. Every two or three days, the entire set of modems at a remote site would go down in the middle of the night. About two hours later, they would mysteriously come back up by themselves. The reason? It turned out that a cleaning person was unplugging the modem sets to plug in a vacuum cleaner. Without being able to study the results of automated tracking, this may never have been solved!

Paper logging isn't a very sophisticated tool for the Xyzzy analysts and programmers who maintain the software systems. For example, if several users have said that a certain report constantly needs to be re-run, detailed analysis may be necessary to recognize the underlying problem and a broader solution. The more thorough the tracking system, the more helpful the analysis available to the programmers.

Proper tracking helps companies be pro-active, which is much more cost-effective than being reactive. Let's take, for example, Canada Post. When Robelle moved to a newly constructed office, mail addressed to us was suddenly undeliverable. Canada Post hadn't had the foresight to assign the new building to any of their routes. This was only accomplished some months later after much struggling with reels of red tape. In the meantime, we had to drive several miles to a postal station *every day* to pick up our mail. If Canada Post had been pro-active, assigning a route as the building was constructed, instead of reactive, it could have saved us both time and money. I guess they're not reading their route analysis reports.

The Xyzzy Company could also benefit from having an automated system in place for detecting potential problems, and by analyzing the reports effectively.

Technical / Software Support

For company Xyzzy, the technical and software support help-desks are similar in nature to those for operations. The difference here is that some of the techies *are* the programmers. Trouble calls can be answered quickly, and fires put out as they flare up. Programming and procedural changes can be made to fix the problems at hand.

The non-programmer techies have a paper-shuffling memo system in place. They produce work orders, make notes made on a white-board, and distribute documents to programmers who need to see the bugs/enhancement requests. The techies work on PCs, creating their documents using Nerd-Perfect, the company standard word processor.

One good thing about this system is its responsiveness. The caller can get quick feedback regarding a problem, and possibly a fix right away. Another good thing is that the documentation created is stored on a file server, and scanned by others on the network if required. Tape backups are stored off-site.

There are some disadvantages, however. If systems for tracking bugs and enhancements are not put into place, the tendency is for the latest gripe to be acted upon instead of the most important. After hanging up the phone from a support call, the programmer may have the solution done in his head. However, when dealing with large software systems, the programmer may not have "the big picture" available. His quick solution may become an arena for problems in other code that he is not responsible for.

Another area for improvement is the document system. Although the Xyzzy documents are kept electronically and in one place, it does not force a dialog with other techies, programmers, and analysts. Because a document "magically" appears, other interested parties may not even notice it. The use of electronic mail, a bulletin board, or even a file with a list of topics and keywords indexed to the document files, would be an improvement.

By using these suggestions, Xyzzy could improve its technical support, and also reduce its research and development cycle.

Customer Support

Although Xyzzy manufactures a wonderful product, widget-removers, the pride of the company is its customer support. Created internally, this customer tracking system rivals (and may even surpass) Hewlett Packard's. Fully on-line and automated, a customer information database is kept current and checked for data integrity on a regular basis. Other electronic systems, such as invoicing, materials management, shipping, inventory and order-desk, are seamlessly joined together.

When a customer-support person answers a client call, all of the information for the client they are talking with is a few keystrokes away. The on-line inquiry system is made efficient by the use of indexing. Partial name look-up and sound-alike algorithms facilitate finding customer information quickly.

Support personnel use a problem log database that is inspected periodically by ad-hoc and scheduled reports. These provide Xyzzy's management with an essential tool for analyzing and reacting to customer feedback.

In this case, Xyzzy excels. Even I'm envious.

The Robelle Scenario

Robelle's work style most likely differs from most companies, but some of the things that we do can be adapted at other shops.

Most of Robelle's programmers work mainly from their homes. They have good PCs, fast modems into the HP 3000, and reference manuals or CD-ROM access to manuals. Their commute time is low, and productivity high due to working in relative isolation. I say *relative*, as everyone has easy access to other people via electronic mail.

Help, and occasionally just good conversation, is a bit-burst away.

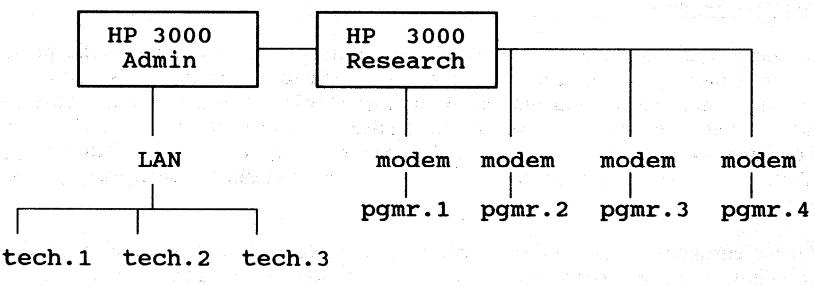


Figure 1: Robelle's Network (Simplified)

Figure 1 is a simplified view of Robelle's HP 3000 network, showing how the programmers and techies are attached to the HP 3000s. The Admin computer runs all of the Management Information System (MIS) programs, as well as the customer call-tracking software. The Research computer is the "crash-and-burn" machine, where all development and testing is performed.

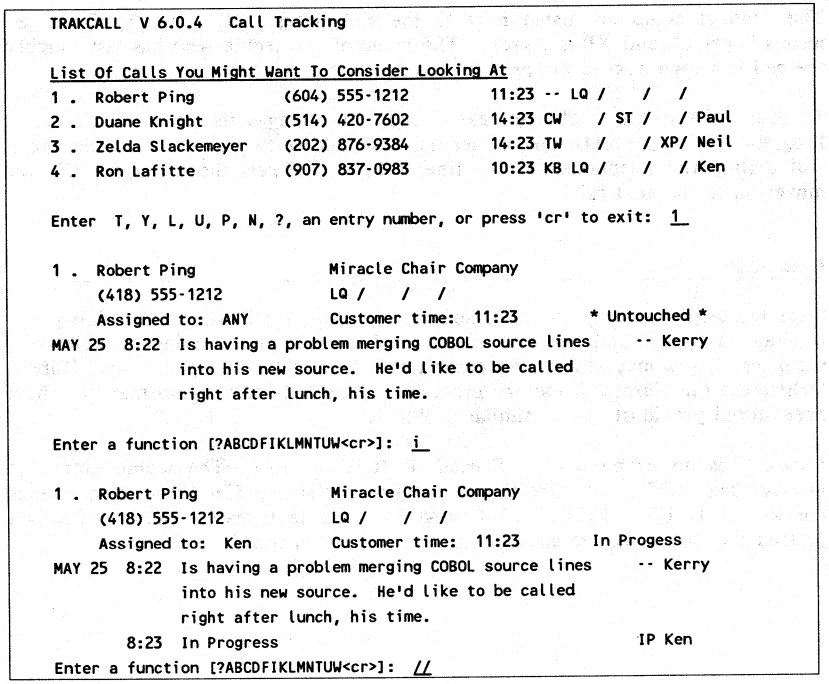
Although the machines are physically next to each other, the programmers work off-site from their homes. This prevents the technical support staff in the office from calling out questions to the programmers and bothering them. They must use electronic mail to contact them, or, occasionally, the telephone.

Xpress, an electronic-mail package developed and marketed by Robelle, is used within the company to communicate between the techies, administrative staff, sales staff, and programmers. One favorite feature of Xpress is the "Auto:- Tell" option. A Tell or Warnf message can appear on the screen when users receive mail. Robelle takes advantage of Xpress' multi-machine capability of transferring messages and data, as well as its programmatic and gateway access to other networks.

The on-line techies are the front-end of any technical support system. At Robelle, they answer the telephone, usually on the first ring. If the phone rings more than twice, the techies are probably busy and the caller chats with a sales or administrative person. This is an advantage for the customer - the call will not be lost. Using the computer, the sales or admin person quickly looks for the client in the customer database. Then, if the techies are still busy, the call is logged into a call-tracking system, complete with a short description.

A reminder pops up on the techies' terminals every five minutes to let them know that a new technical call was logged. After completing their current task, they look at the call log to see if the customer should still be called. Another techie may already be handling the call, or the customer may have left a message that they'd like to be called at a later time.

Once in touch with the customer, the techie may be able to solve the problem right away. However, if research has to be done, or a direct question to the programmer is required, the call is logged into another tracking system.



```
TRAKCALL  V 6.0.4   Call Tracking

List Of Calls You Might Want To Consider Looking At
1 .  Robert Ping           (604) 555-1212          11:23 -- LQ /    /    /
2 .  Duane Knight          (514) 420-7602          14:23 CW    / ST /    / Paul
3 .  Zelda Slackemeyer     (202) 876-9384          14:23 TW    /    / XP/ Neil
4 .  Ron Lafitte           (907) 837-0983          10:23 KB LQ /    /    / Ken

Enter  T, Y, L, U, P, N, ?, an entry number, or press 'cr' to exit:  1

1 .  Robert Ping                Miracle Chair Company
     (418) 555-1212             LQ /    /   /
     Assigned to:  ANY          Customer time:  11:23        * Untouched *
MAY 25  8:22  Is having a problem merging COBOL source lines       -- Kerry
              into his new source.  He'd like to be called
              right after lunch, his time.

Enter a function [?ABCDFIKLMNTUW<cr>]:  i

1 .  Robert Ping                Miracle Chair Company
     (418) 555-1212             LQ /    /   /
     Assigned to:  Ken          Customer time:  11:23        In Progess
MAY 25  8:22  Is having a problem merging COBOL source lines       -- Kerry
              into his new source.  He'd like to be called
              right after lunch, his time.
        8:23  In Progress                                          IP Ken
Enter a function [?ABCDFIKLMNTUW<cr>]:  //
```

Figure 2: The Call Tracking Software

TRAKCALL is the name of the telephone call tracking software at Robelle, and part of its functionality is shown in *Figure 2.* Calls are logged using the same program with a different entry point, and the information is stored in an IMAGE date base. The keyed-access field is the telephone number of the customer.

When the techie runs or re-activates TRAKCALL, a list of all of the entries in various stages are presented. The status of the call is shown next to the customer time. For example, in *Figure 2*, the status of the first call is "--", indicating that nobody has yet taken this call. The CW and TW status codes on the next two lines mean "Customer Wait" and "Techie Wait". Tech support often waits for a customer to reproduce the results of some problem, or to send details by fax. When a call is in "Techie Wait" status, a techie is currently working on the solution for that call.

The KB status code indicates that the call is waiting to be entered into Robelle's other tracking system, Karnak B. We'll see shortly how Karnak B provides a forum for dialog between all techies and programmers, as well as being a product knowledge database.

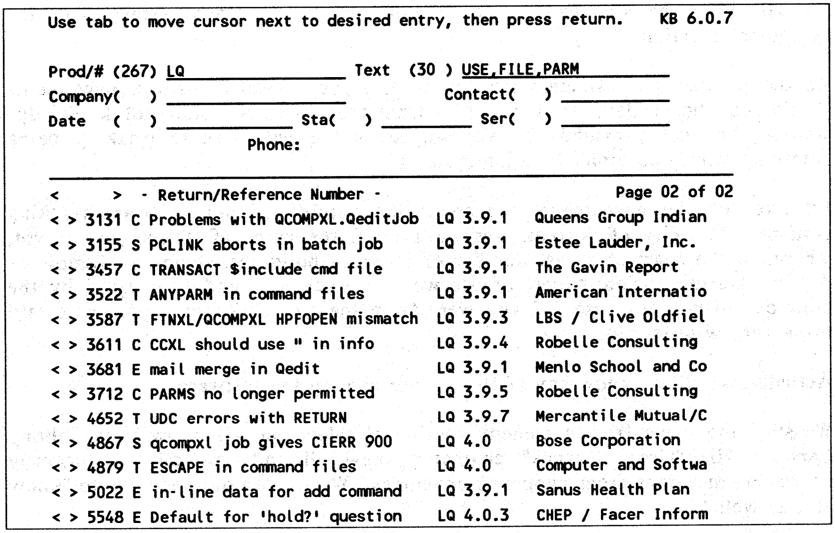
The product codes are listed next to the call status. LQ indicates Qedit, ST means Suprtool, and XP is Xpress. The name of the techie who has last touched the call is shown next to the product codes.

As soon as a techie is able to take a call, she changes its status to IP, or "In Progress". This prevents another techie from taking the same call while she is still dialing the telephone. If the line is busy, she sets the status to BZ, and moves on to the next call.

Karnak B

Once the techie has engaged the client in conversation ("Really? It's raining here in Vancouver..."), and discovered the problem ("You're having trouble with use-files?"), she may realize she doesn't know the solution right away. Robelle techies use the Karnak B Inquiry program to seek out information that may have been stored previously about similar problems.

Figure 3 is an example of a Karnak B (KB) inquiry. The techie entered a product code of LQ (for Qedit) into the product search field, and then entered the keywords USE, FILE, and PARM into the text search field. After she pressed the Return key to start the search, KB found thirty entries.



```
Use tab to move cursor next to desired entry, then press return.    KB 6.0.7

Prod/# (267) LQ________________ Text  (30 ) USE.FILE.PARM_________
Company(   ) ___________________          Contact(    ) __________
Date   (   ) _____________ Sta(    ) _________ Ser(    ) __________
                     Phone:
______________________________________________________________________
<     >  - Return/Reference Number -                      Page 02 of 02
< > 3131 C Problems with QCOMPXL.QeditJob  LQ 3.9.1   Queens Group Indian
< > 3155 S PCLINK aborts in batch job      LQ 3.9.1   Estee Lauder, Inc.
< > 3457 C TRANSACT $include cmd file      LQ 3.9.1   The Gavin Report
< > 3522 T ANYPARM in command files        LQ 3.9.1   American Internatio
< > 3587 T FTNXL/QCOMPXL HPFOPEN mismatch  LQ 3.9.3   LBS / Clive Oldfiel
< > 3611 C CCXL should use " in info       LQ 3.9.4   Robelle Consulting
< > 3681 E mail merge in Qedit             LQ 3.9.1   Menlo School and Co
< > 3712 C PARMS no longer permitted       LQ 3.9.5   Robelle Consulting
< > 4652 T UDC errors with RETURN          LQ 3.9.7   Mercantile Mutual/C
< > 4867 S qcompxl job gives CIERR 900     LQ 4.0     Bose Corporation
< > 4879 T ESCAPE in command files         LQ 4.0     Computer and Softwa
< > 5022 E in-line data for add command    LQ 3.9.1   Sanus Health Plan
< > 5548 E Default for 'hold?' question    LQ 4.0.3   CHEP / Facer Inform
```

Figure 3: Karnak B Inquiry

Various status codes appear next to the KB reference numbers in *Figure 3.* These codes are as follows:

E	Enhancement Request
C	Closed
M	Mystery
P	Send a Pre-Release tape to the customer
T	Pre-Release tape has been sent
B	Bug
O	Open

Figure 4. KB Status Codes

The product code and version number are displayed on each line of the list, as well as the name of the company from where the call came. Using the tab key, the techie may move the cursor to the entry that he would like to scan. KB then displays the free form text and the techie may page forwards and backwards through it, or print the entire entry.

The Karnak B Story

The original idea to keep track of Robelle's customer problems and enhancements is accredited to Mike Shumko. In 1988, Mike was the only person on Robelle tech support. Although being very knowledgeable about Robelle's products and the HP 3000, he still had to pester Bob Green and David Greer with the occasional question.

As the number of customers steadily rose, Robelle realized that a new techie would soon be needed. How better to train them, if a repository of knowledge were to be readily available? And so, Karnak B was born (Karnak A being Administration's customer tracking system).

I'd like to take a moment to recount the naming of our software tracking systems. The city of Karnak was built upon the ruins of Thebes, in Egypt, where, in the heart of a vast number of religious buildings, stood the Temple of Amon. During ancient times, priests were also scholars, and considered by the populous to be very wise. If you were from the city of Karnak, you were said to be very wise indeed.

Actually, we didn't know any of this before naming the software.

We stole the name from a comedy routine of television talk-show host, Johnny Carson. His "Great Karnak" character knew all and saw all, mysteriously answering questions from unopened envelopes. We wanted our software to "know all", as well.

Karnak B is used world wide in read-only mode by Robelle distributors. An update of the software and the database is sent to each distributor every few months. This provides them with "automated" tech support from Robelle. By browsing the entries created after a certain date, the distributors can find out which new problems have occurred, and what we are currently working on. Here is an excerpt from the Karnak B manual that describes KB.

> *Karnak B consists of an IMAGE database filled with interesting information about Robelle products and HP 3000 related topics. On-line inquiry is performed via the KBINQ program. Techies enter messages into Xpress, Robelle's e-mail package, and mail them to the Karnak B user. A background process, KBLOAD, receives the free-format messages and loads them into the KB database.*
>
> *Using Omnidex, a third-party IMAGE indexing tool, KBINQ allows online inquiry of the KB database by multiple and partial keywords.*
>
> *KBINQ has been designed with speed in mind - speed in display, type-ahead usability, and speed in retrieval. On our system, a lookup qualifying 1000 entries requires about 5 seconds, and one with 400 entries appears almost instantaneously. We are more concerned with narrowing our search down to a few entries, as we couldn't possibly browse 1000 entries comfortably.*

Robelle Dialog

When a techie cannot find a solution in Karnak B, or <gasp!> when a bug is discovered and must be reported, then the other role of Karnak (KB as a dialog tool) is put into action.

The dialog between the techies and the programmers at Robelle is very important, as much of Robelle's product development is based on user feedback. This ear-to-the-ground approach is greatly facilitated by the Karnak B system.

In fact, many of Robelle's customers are aware of Karnak B, and often call to add their personal desires to the "wish lists". Then, when their enhancements or bug-fixes are done, a product pre-release tape may be sent to their site, accompanied by a letter from Karnak B detailing the results.

Initiating the KB Dialog

To start the KB dialog, a techie submits an entry to be loaded into the KB database, using a standard template. Here's an example:

```
Company:  Acme Anvil Company
Caller:   Wile E.Coyote
Phone:    (800) 582-1700
Serial#:  37E
Product:  QX 4.1
Ref:      NEW
Status:   E
Techie:   Paul Gobes
Line1:    User would like Qedit/UX to support the
Line2:    mouse.
Comment:  You never know...

Wile types with only his left hand, using the mouse
with his right.  He'd like Qedit/UX to be able to read
the mouse movements.
```

Figure 5. Karnak B Raw Entry

To help identify the nature of the entry, the techie assigns one of the KB status codes. A background process, KBLOAD, sits waiting for mail from techies all day long. When it receives an entry, KBLOAD scans it for syntax, and then loads it into the database. Keyword indexes from the free form text and keyworded fields are created, and replies are mailed to anyone whose product filters are set to receive that product's entries. For example, Bob Green does not need to see Suprtool reports, so one of his exclude-type filters is set to product code ST.

When a programmer reads the mail from Karnak B, he replies to the message, according to his actions. He may change the status code, for example, from

"Bug" to "Send a pre-release tape" after having fixed the problem.

Robelle's internal MIS programs are tracked in a similar manner. Instead of talking to techies, however, the end users submit bug reports and enhancement requests directly to Karnak B. Karnak B sends the MIS programmer an e-mail message, informing him of the problem. When a bug is fixed or an enhancement made, the MIS programmer appends a note to the entry in KB, containing an install date. The end users are then mailed a message from Karnak B, announcing the new status of their request. End users like this quick, "personalized" feedback, the final "conversation" in the KB dialog.

Example Karnak B Entry

The following example is a typical entry in Karnak B illustrating how the the KB dialog develops between the techies and the programmers. Each * *Note* line indicates the start of an appended message.

```
Ref:          5465
Company:      Blingblang Motor Company
Caller:       Annette Schwartz
Phone:        (800) 555-1212
Serial#:      88L
Subject:      justify right numbers on list
Line1:        Listing numerics should be Right Justified
Line2:
Product:      ST 3.4
Techie:       PAUL GOBES
Status:       Closed
```

* *Orig: PAUL GOBES 15 Sep92 9:36 AM* *
Earl A. Sillavan of Blingblang would like the List command to not LEFT Justify numerics but instead Right justify them.

* *Note: (E:E) DAVID GREER Where should the sign go? 15 Sep92 10:12 AM* *
If we right-justify numbers, should the sign come after the number (like output xxx,ascii) or before the number?

* *Note: (E:E) PAUL GOBES Sign s/b after the number 15 Sep92 11:08 AM* *
I think it should be after, it's clearer.

* *Note: (E:E) DAVID GREER What does the user say? 16 Sep92 8:23 AM* *
Paul, I really want to know what the customer thinks.

* *Note: (E:E) PAUL GOBES user wants trailing sign 22 Sep92 10:55 AM* *
I phoned the user and they requested trailing signs.

* *Note: (E:P) NEIL ARMSTRONG Done in ST 3.4.06 23 Feb93 10:20 PM* *
The list command now defaults to right justification in the listing of numeric fields. Send pre-release.

* *Note: (P:O) NEIL ARMSTRONG no longer on service   23 Feb93 10:24 PM* *
Blingblang is not currently on service. I will check with Jennifer to see if they intend to go back on service.

* *Note: (O:C) NEIL ARMSTRONG send ST 3.5   08 Mar93 2:11 PM* *
Blingblang will receive ST 3.5 when they pay their maintenance. Jennifer says that it is in the works. Call closed. I have told them about right justification.

Figure 6. Karnak B Dialog

Karnak B e-mails each addendum to all of the techies and programmers, so that an ongoing conversation is formed. Later, anyone can read the full text of the interaction using KBINQ.

The Wrap-Up

How Does Robelle Tech Support Rate?

Robelle may not be doing everything right yet, but it seems that in some areas, we're getting close. Judging by customers' comments such as, "*I like the easy, breezy, free-wheeling feeling from tech support*", I think that in this important area at least, we are on the right track. Here again are the main goals of tech support, followed by how Robelle tries to achieve them:

☐ **Provide a quick and inexpensive solution for the client**
We publish our tech support phone number in all of our product documentation, and our techies really do try to answer the phone call on that first ring. The techies are very knowledgable, providing everything from quick bug workarounds to a little hand-holding for those difficult tasks.

☐ **Track the problem call**
Robelle tracks trouble calls using the TRAKCALL and Karnak B software packages.

☐ **Create a dialog between technical staff and programmers**
Using electronic mail coupled with Karnak B, Robelle techies and programmers converse and problem-solve within the HP 3000 "cyberspace".

☐ **Log the problem and the solution**
We log all of the products' problems and solutions into the KB database. Several batch jobs provide reporting functions to aid in determining cyclic problems.

What You Can Do Right Now

Instead of thinking that you must jump in and purchase a new problem tracking system, hire more techies, get electronic mail, and write in-house reports, start by implementing a step-by-step help-desk solution.

It is always advisable to keep long-term goals in mind while designing your help-desk, but start small. *Aim to satisfy some of your customers' needs now.*

Simply logging calls on-line to a flat file may be a good first step. You can use some clever command files to do this, either through EDIT/3000 or with Robelle's Qedit. Create a command file that adds entries to a flat file with a date/time stamp, as well as the name of the logger. Another command file could find entries based on keywords and smart pattern matching.

Actually, I do think that everyone should have electronic mail. Its benefits far outweigh the cost - which is mostly in end user education. Some electronic-mail packages let you hook into Internet - the largest network in North America. As a bonus, you can probably find answers to some of your questions there.

Of course, as a very *first* step, you have to answer the darn telephone.

Ken Robertson is the manager of MIS at Robelle Consulting. Along with his users at Robelle, he has designed and implemented the product tracking systems. They are not currently for sale.

Copyright Robelle Consulting Ltd. 1993

Permission is granted to reprint this document (but not for profit), provided that copyright notice is given.

Paper No. 5025

A Prototype for an MPE/iX Menu based User Interface

Mark Farzan
Santa Fe International Corp.
1000 S. Fremont Ave.
Alhambra, Ca. 91803
Phone: (818)300-6630
(818)342-6295

You are the system manager for a medium-size company with a few remote sites, and a Local Area Network. As you work with your favorite editor on the latest fix for a utility program, the phone rings. It is your contact in the Istanbul office. He is having a problem with a just installed, state-of-the-art spooler package. It sounds simple to resolve, but you might as well long on to their system, and check it out. May be it is short on disc space, or a few files need to be moved around. You press the break key, and your terminal displays a selection of various frequently used applications, and tools. You select the remote access menu, and with a few keystrokes, you are logged on the Istanbul office. After taking care of the problem, you log off and are then transferred back to the menu. You select the editor and you are back to where you left.

The problem

System managers, who manage several hosts and products are frequently interrupted to provide support services to their users. To do so, they are switching among hosts, accounts, groups, and applications several times a day. This constant switching is nothing new to system managers, or to users. It is done to overcome the following traditional MPE limitations:

Terminal datacomm interface (one session per physical port)

Account-dependent applications (e.g. finance application can only be run from FIN account only)

Inability to run additional applications while in BREAK.

Cross-Account Security (e.g cannot transfer files across accounts.)

Among the results of this periodic account and host switching is increased overhead in system resources, confusion for the user, left-over files from one system to the other, and more.

Solutions

Solutions to part of the above problems have been available from time to time. In 1988, a suggestion for MPE/V was to patch system intrinsics to allow process switching while in BREAK (suggested by M. Kohon). Then came purely menu products such as SELECT from Robelle's QLIB library, and later on SECURITY/3000's menu interface from VeSoft. Recent Windows look-alike products like WINGSPAN by Software Research Northwest, ENTERFACE by Enterface Inc., and MENUMASTER/3000 by Industrial Management Systems, have made excellent contributions to the terminal users. The HOTKEY/3000 by Riviera Software allowed true process switching in 1990. The introduction of HP's DTC (Datacommunications and Terminal Controller) facilitated switching sessions but in a limited sense. The introduction of WINDOWS for MS-DOS, and multiple HP3000 sessions using emulators under WINDOWS made this problem almost disappear for the PC users, but for terminal users the problem still exists.

In this article a simple, yet affordable, method of accomplishing process switching using the BREAK key is suggested. This technique uses the NS LOOPBACK Network Interface, and when combined with conventional benefits of using menus and menu processors in general, such as common interface, savings in keystrokes, application security, ease of use, audit and traceability, allows system administrators to define powerful user interfaces. And it provides users, and system administrators with maximum productivity, and minimum overhead in system resources.

NS/3000's LOOPBACK Service

The Network Services LOOPBACK interface is a function of HP's NS/3000 product that lets you establish a remote session from one account in a node to another account in the same node. This interface can be used effectively for varieties of applications such as:

> Transferring files across accounts without doing a RELEASE to the source file, logging to the target file, and pulling the file.
>
> Running applications in different accounts.

This interface is available as a NETCONTROL START option, and takes a few minutes to configure using the NMMGR utility in the MPE/V, and MPE/iX.

To show this feature let's assume the local node name for your host is PLUTO. If the LOOP network interface is enabled, then the user would simply type DSLINE PLUTO, followed by the "REMOTE HELLO ...." to create another session in the same host. To make this session be easily recognizable you can also assign a new ENVIRONMENT ID to this "node" as follows:

```
DSLINE FINANCE=PLUTO
REMOTE
FINANCE# Hello auditor,user.FIN
```

See your NS/3000 manual for more information and the detailed syntax.

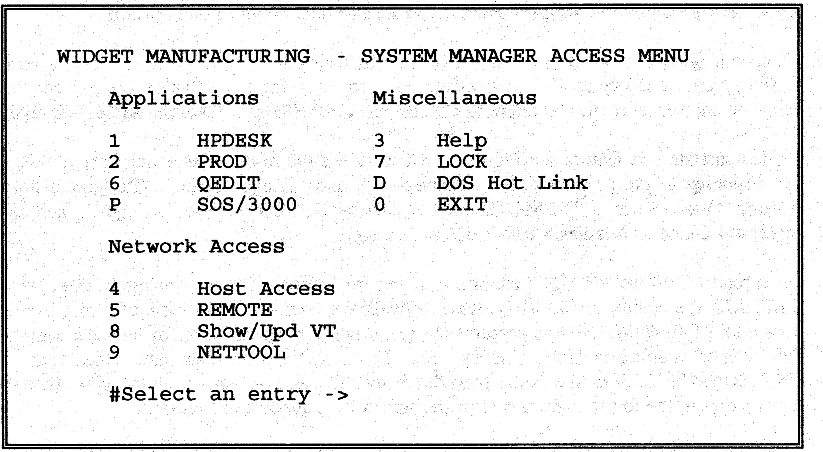
The BREAK key:

Very little documentation is available about this important key on your terminal, but anyone who has tried to run a program while in BREAK mode, is familiar with the error message "ABORT (YES/NO)?, and its subsequent "COMMAND NOT ALLOWED IN BREAK.." message. This is because MPE allows only a subset of MPE commands, to be executed while in BREAK; which in effect makes the BREAK key as it exists today a rather useless key, unless you do something like :REMOTE HELLO to another machine. How about the same machine ? That's right, use the same node name and voila, you just created another session from the same terminal.

The menu shown in Figure 1, is an implementation of the above principles. For simplicity we discuss only 3 typical cases: Suspending an e-mail program, accessing a remote host, and switching to DOS from within HP3000.

CASE 1: Suspending an Electronic Mail program.

A very useful example of the concept thus far developed is the ability to logon to your e-mail program once using a separate LOOPBACK session, stay logged on as long as you want, and simply BREAK out of it, or RESUME back into it to do other tasks. Then you can use a simple background menu to hide all the condition checking, parameter parsing, and keystrokes saving, and you have got yourself something like a juggler that does not drop any balls!. Figure 1 is a working example of the top level menu. The menu is displayed using the command file that appears as listing 1.



```
WIDGET MANUFACTURING  - SYSTEM MANAGER ACCESS MENU

    Applications          Miscellaneous

    1      HPDESK         3    Help
    2      PROD           7    LOCK
    6      QEDIT          D    DOS Hot Link
    P      SOS/3000       0    EXIT

    Network Access

    4      Host Access
    5      REMOTE
    8      Show/Upd VT
    9      NETTOOL

    #Select an entry ->
```

Figure 1: Main Menu

By selecting option "1" the command file tests for the following sequence of conditions, and performs the appropriate actions:

o If the variable DESKHOSTON is "false", it initiates a remote LOOPBACK session to the host containing the HPDESK program and sets this variable to "true" on successful connection for next time around.

o If the program was in BREAK mode then a RESUME to activate the suspended program would not fail. This condition is tested using the statement REMOTE RESUME, and then checking for CIERR 1686.

o If not in BREAK mode, then BUILD a REMOTE temporary file DESKOFF. The presence and absence of this file in the remote node is used to set a local variable DESKON accordingly. As far as I know this is the only way to fake setting and inquiring a remote variable (especially if the remote is MPE/V.)

o If DESKON is "true" then we are just returning to top menu after pressing the BREAK, otherwise it is assumed that the program was never ran, so we issue the REMOTE HPDESK to initiate the HPDESK command file (or UDC.)

Also, a few cosmetic-related statements are done to prevent undesirable displays from appearing on the terminal.

Using this technique all the related databases and files remain open and there is no overhead of opening and closing the user trays after pressing the BREAK, since the user context is unaltered.

CASE 2: Connection to Remote Hosts, and Dynamic Command Generation

Another technique detailed in the command file in listing 1, is the facility to create a remote session, transfer the control to that session, and on returning from that session, dynamically create an abbreviation "Rxx", where "xx" is the ENVIRONMENT ID of the REMOTE session.

To demonstrate this feature see Figure 2, which shows the result of selecting option "4", and the responses to the prompts "#Host Name -->", and "#Logon String". The command file (Listing 1) generates a "REMOTE :<hostname> HELLO <logon string>", and upon successful connection issues a REMOTE to that node.

Upon return from the REMOTE command, either from terminating the session, or even causing a BREAK, the command file XEQs the SHOWENVX command file (Listing 2) that basically does a DSLINE @;SHOW and captures the result into a temporary file for later massaging by SHOWENV command file (Listing 3). The SHOWENV command file uses the ENVIRONMENT ID of the node, precedes it by "R", and displays it along with other vital information in the lower right corner of the screen as it appears in Figure 3.

A maximum of 20 remote simultaneous sessions can be established using the MENU listed in Listing 1, but screen scrolling changes would be required.

CASE 3: Accessing DOS from the menu

Another benefit of using such a menu is the ability to access the DOS shell if running emulators like Advancelink, Reflection, or MiniSoft. Choice "D" in the menu sends the appropriate escape sequence to the terminal to execute the COMMAND.COM; obviously other programs could be executed by ECHOing the statement "[ESC]&oC&>[DOS prog]" to the terminal running under advancelink, or "[ESC]&oCSHELL [DOS prog]" for Reflection. Contact your vendor for exact syntax, and description of the Host initiated commands.

A good use of this feature could be to link your menu system on the HP3000 to a menu driver running under DOS.

Limitations

The prototype discussed thus far obviously has its limitations. First the application being executed must have BREAK enabled. This does not seem to be a problem for a majority of applications, but it is a factor that must be considered when using purchased applications. Second, the technique described for the HPDesk program (i.e its detection of whether it is in BREAK or not, etc..) must be coded for any application requiring switching. This may tend to make the menu command file unnecessarily large. In our example the HPDesk program is the only one automatically switchable, the other applications, (e.g PROD, QEDIT, SOS) although not coded for switching, they are still switchable, manually using the BREAK, and RESUME commands.

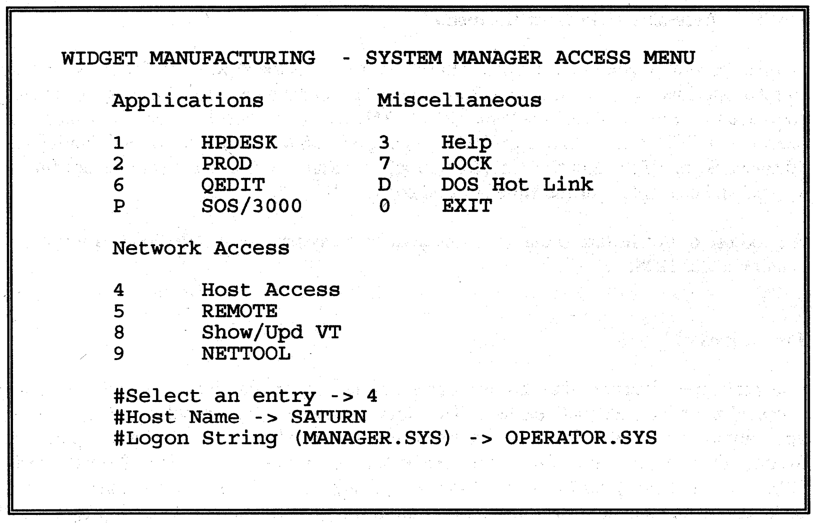
Conclusions

Using the NS/3000 LOOPBACK Network Interface, the BREAK key, and a simple command file, a simple yet affordable and powerful user interface can be built for terminal and PC users alike. This technique provides process switching in the same host, allows multiple host connections, reduces the system overhead associated with multiple logons, and it improves user productivity. Combined with techniques in logon and access control, and taking advantage of the powerful MPE/iX command interpreter structures, this method can be used to integrate user interfaces totally.

References:

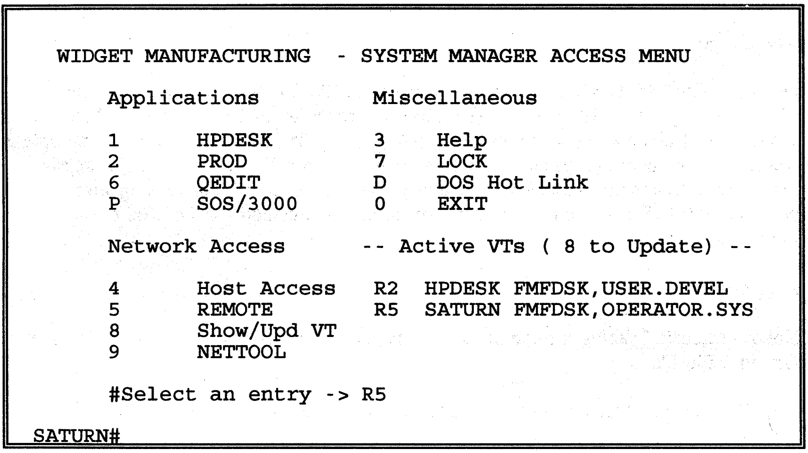
Kohon, Michel; "Taking a Short Break..."; Paper No. 175;Proceedings of the 1988 Interex; Orlando, Florida.

NS/3000 Network Manager Reference Manual, Vol 1; HP Part# 32344-90002.



```
WIDGET MANUFACTURING  - SYSTEM MANAGER ACCESS MENU

    Applications         Miscellaneous

    1      HPDESK        3    Help
    2      PROD          7    LOCK
    6      QEDIT         D    DOS Hot Link
    P      SOS/3000      0    EXIT

    Network Access

    4      Host Access
    5      REMOTE
    8      Show/Upd VT
    9      NETTOOL

    #Select an entry -> 4
    #Host Name -> SATURN
    #Logon String (MANAGER.SYS) -> OPERATOR.SYS
```

Figure 2: Main Menu - Remote Access Selection



```
  WIDGET MANUFACTURING  - SYSTEM MANAGER ACCESS MENU

      Applications         Miscellaneous

      1      HPDESK        3    Help
      2      PROD          7    LOCK
      6      QEDIT         D    DOS Hot Link
      P      SOS/3000      0    EXIT

      Network Access      -- Active VTs ( 8 to Update) --

      4      Host Access   R2  HPDESK FMFDSK,USER.DEVEL
      5      REMOTE        R5  SATURN FMFDSK,OPERATOR.SYS
      8      Show/Upd VT
      9      NETTOOL

      #Select an entry -> R5

SATURN#
```

Figure 3: Main Menu - Dynamic Command Creation

```
OPTION nolist
echo #Initialization
setvar esc, chr(27)
setvar cursup, "!esc"+"A"
setvar home, "!esc"+"H"
setvar clear,"!esc"+"J"
setvar hcl, "!home"+"!clear"
setvar desknode,"JUPITER"
setvar choice, ""
setvar host, ""
setvar skiphcl,0
setvar hstcntl, "!esc"+"&oC&"
setvar hpdeskon,FALSE
setvar deskhoston,FALSE
setvar env_cursor, "!esc"+"&a 039c "
setvar env_dic &
"R1 R2 R3 R4 R5 R6 R7 R8 R9 R10R11R12R13R14R15R17R18R19R20"
while choice <> "0"  DO
   if skiphcl=0 then
   ECHO !hcl
   ECHO
   ECHO    WIDGET MANUFACTURING  - SYSTEM MANAGER ACCESS MENU
   ECHO
   ECHO                  Applications        Miscellaneous
   ECHO
   ECHO               1      HPDESK         3    Help
   ECHO               2      PROD           7    LOCK
   ECHO               6      QEDIT          D    DOS Hot Link
   ECHO               P      SOS/3000       0    EXIT
   ECHO
   ECHO                  Network Access
   ECHO
   ECHO               4      Host Access
   ECHO               5      REMOTE
   ECHO               8      Show/Upd VT
   ECHO               9      NETTOOL
   ECHO
if bound(env_id_1) then
echo ![esc]&a039c011R-- Active VTs  (Select 8 to Update) --
   setvar temp 1
   setvar nxt_row, 13
   while temp<= indx do
   echo !env_cursor![nxt_row]RR![env_num_!temp]   &
        ![env_id_!temp] ![env_logon_!temp]
   setvar nxt_row nxt_row + 1
   setvar temp temp+1
   endwhile
echo ![esc]&a016c019R
endif
   INPUT choice,"                #Select an entry -> "
   else
```

Listing 1: Top Level Menu (MENU.CMDFILES), continued

```
setvar skiphcl,0
input choice
endif
setvar choice,UPS("!choice")
If choice = "1" THEN
   echo !hcl
   if not deskhoston THEN
      echo #Connecting to HPDesk Host
    REMOTE :HPDESK=!desknode hello &
           !hpjobname,!hpuser.!hpaccount
      FILE DESKOFF=DESKOFF:HPDESK
      setvar deskhoston,TRUE
   endif
   continue
   REMOTE setjcw cierror=0
   REMOTE :HPDESK  RESUME
   REMOTE if cierror=1686 then
   echo !cursup!cursup!clear
   comment *** not logged on or just logged off ***
   REMOTE   continue
   REMOTE   build DESKOFF;TEMP
   REMOTE   setjcw cierror=0
   REMOTE endif
   echo !cursup!cursup!cursup!clear
   setjcw cierror=0
   continue
   purge *DESKOFF,TEMP
   if cierror = 0 then
      setvar hpdeskon,false
   endif
   if not hpdeskon THEN
      setvar hpdeskon,TRUE
      REMOTE HPDESK !hpjobname
   endif
elseif choice = "2" THEN
   echo !hcl
   PROD
elseif choice = "3" THEN
   echo !hcl
   HELP
elseif choice = "4" THEN
       input host,"                #Host Name -> "
   setvar deflogons,"!hpuser.!hpaccount,!hpgroup"
   setvar logonstr,"!deflogons"
   input logonstr,"          #Logon String (!deflogons) ->"
   echo !hcl
   setjcw cierror = 0
   continue
   REMOTE :!host HELLO !hpjobname,!logonstr
   if cierror <> 0 then
```

Listing 1: Top Level Menu (MENU.CMDFILES), continued

```
      Echo
      Echo Host access failed
      input resp, "Enter any key to continue"
   else
      setjcw cierror = 0
      continue
      REMOTE
      echo #Updating VT list now...
      XEQ SHOWENVX.CMDFILES.SYS
   endif
elseif choice = "5" THEN
   echo !hcl
   setjcw cierror = 0
   continue
   REMOTE
   if cierror <> 0 then
      input resp, "Enter any key to continue"
   endif
elseif choice = "6" THEN
   echo !hcl
   continue
   QEDIT
elseif choice = "7" THEN
   echo !hcl
   continue
   LEAVE
elseif choice = "8" THEN
   XEQ SHOWENVX.CMDFILES.SYS
   setvar resp 'N'
   input resp, "Enter ENV# to goto:  "
   if numeric("!resp") then
      REMOTE !resp
   endif
   input resp, "Enter any key to continue"
elseif choice = "9" THEN
   NETTOOL.NET.SYS
elseif choice = "P" THEN
   SOS3K
elseif choice = "D" THEN
   setvar doscmnd,"COMMAND"
   ECHO !hstcntl!>!doscmnd
elseif choice = "REDO" then
   echo REDO Feature not available yet.
   echo Use line modify at line below
   echo !oldchoice
   setvar skiphcl,1
elseif pos("!choice","!env_dic") > 0 and choice <> "0" then
   setvar env_choice &
     str("!env_dic", pos("!choice","!env_dic")+1,2)
   REMOTE !env_choice
elseif choice <> "0" then
   continue
```

Listing 1: Top Level Menu (MENU.CMDFILES), continued

```
      !choice
      input resp, "Enter any key to continue"
      setvar oldchoice,"!choice"
      setvar choice,""
   endif
ENDWHILE
```

Listing 1: Top Level Menu (MENU.CMDFILES)

```
comment
comment do a dsline @ and send the output to the DSLOUT
comment
if finfo("dslout",0) then
  Purge dslout,temp
endif
file dslout;rec=-80,32,f,ascii
DSLINE @;SHOW > *dslout
RUN ci.pub.sys;info="XEQ SHOWENV.MAINT.SYS DSLOUT" ;parm=3 &
    ;stdin=DSLOUT
```

Listing 2: SHOWENVX.CMDFILES

```
parm dsloutf="DSLOUT"
comment Read all the records in the dsline output and
comment extract envnum, envid, node, logon, prompt
comment format  nn    , <=8  , 8   , 26   , 7
comment **** CALLED BY SHOWENVX command file *****
comment *******************************************
comment 1) First build the array
setvar env_flen finfo("!dsloutf",19)
if !env_flen <9 then
    return
endif
setvar env_cnt !env_flen
setvar indx 1
setvar env_rec rpt(" ",80)
setvar env_end FALSE
while env_cnt > 0  and env_end = FALSE
      input env_rec
```

Listing 3: SHOWENV.CMDFILES, continued

```
      if str("!env_rec",13,1) = "#" then
         setvar env_num_!indx   str("!env_rec",18,2)
  elseif str("!env_rec",13,2) = "ID" then
         if str("!env_rec",18,1) = "@" then
            setvar env_end TRUE
         else
         setvar env_id_!indx    str("!env_rec",18,20)
         setvar env_id_!indx &
           str("![env_id_!indx]",1,pos('.',"![env_id_!indx]")-1)
         setvar env_len len("![env_id_!indx]")
         setvar env_id_!indx &
           str("![env_id_!indx]",1,min(7,!env_len)) + &
                rpt(" ",7-!env_len)
         endif
  elseif str("!env_rec",1,10) = "NO GENERIC" then
         setvar env_end TRUE
  elseif str("!env_rec",1,4) =  "NODE" then
         setvar env_node_!indx  str("!env_rec",18,20)
         setvar env_node_!indx &

str("![env_node_!indx]",1,pos('.',"![env_node_!indx]")-1)
         setvar env_len len("![env_node_!indx]")
         setvar env_node_!indx &
          str("![env_node_!indx]",1,min(7,!env_len)) + &
          rpt(" ",7-!env_len)

  elseif str("!env_rec",1,5)  = "LOGON" then
         setvar env_logon_!indx    str("!env_rec",18,20)

  elseif str("!env_rec",1,6) =  "LOGGED" then
         setvar env_logged_!indx  str("!env_rec",18,1)

  elseif str("!env_rec",1,6) =  "PROMPT" then
         setvar env_prompt_!indx  str("!env_rec",18,8)

  elseif str("!env_rec",1,1)  = " " then
         setvar indx indx+1
      endif
      setvar env_cnt env_cnt-1
endwhile
If env_end then
   setvar indx indx - 1
endif
comment Now report the results
comment
echo ENV --ENVID-- --NODE-- ------LOGON---------  -ON- -PROMPT-
setvar cnt 1
while cnt <=  indx
```

Listing 3: SHOWENV.CMDFILES, continued

```
  setvar env_outrec &
    "![env_num_!cnt]"        + "  " + &
    "![env_id_!cnt]"         + "    " + &
    "![env_node_!cnt]"       + "    " + &
    "![env_logon_!cnt]"      + " " + &
    "![env_logged_!cnt]"     + "   " + &
    "![env_prompt_!cnt]"
  echo !env_outrec
  setvar cnt cnt+1
endwhile
```

Listing 3: SHOWENV.CMDFILES

A Prototype for an MPE/iX Menu Based User Interface

by

Mark Farzan
Santa Fe International

❒ Typical System Manager's Problems

- Interruptions
- Multiple Hosts
- Multiple Logons

- ❒ MPE Limitations
  - Terminal Interface
  - Application Dependencies
  - Limitations of the BREAK key
  - Cross Account Security

❐ Typical Solutions

- Hotkey/3000 ....(Riviera)
- SECURITY/3000 ....(VeSoft)
- SELECT .... (Robelle)
- WINGSPAN ....(SRN)
- ENTERFACE .... (Enterface)
- MENUMASTER ....(IMS)
- DTC/3000 ....(HP)
- Windows ....(Microsoft)

❒ A NS/CI Based Solution

  - NS/3000 Loopback Service
  - Command File(s)
  - The BREAK Key

❐ The BREAK key and its limitations:

```
:HPDESK
🖐<break>
:BRW
ABORT (YES/NO)?no
COMMAND NOT ALLOWED IN BREAK
```

☐ Using NS/Loop to create new sessions:

```
:HPDESK
 ✋  <break>
:DSLINE FINANCE=PLUTO
ENVIRONMENT 1:FINANCE.SANTA.FE=PLUTO.SANTA.FE
:REMOTE
MPE XL:HELLO AUDITOR,MGR.FIN
HP3000 RELEASE: B.40.00 USER VERSION: B.40.00
MPE/iX HP31900 B.30.45 @ Hewlett-Packard 1987.
FINANCE#BRW
```

- ❐ MPE/iX Command Language
  - An interpretive Prog. Language
  - Setvar, while, if/then/else, input, echo
  - Screen manipulations
  - String functions

☐ Case Studies:

- Suspending HPDesk
- Dynamic Command Creation
- Accessing DOS

❒ Limitations:

  - One session needed per Break
  - Prog. must be BREAK-able
  - Large command file(s)

The END

💣♋❐& ☞♋❐⌘♋■

☎🖰📂🖰✆🗏🗀🗀📫⌛⌛🗏🗀

☎🖰📂🖰✆🗏🗐🗎📫⌛🗎🖭🗄

Accessing TurboImage from Your PC

Paper No: 5026

By: Todd Hubbard and Ed Roden

Crowe Chizek and Company
2100 Market Tower
10 West Market Street
Indianapolis, Indiana 46204-2976
(317) 632-8989

This paper will discuss the use of a client/server tool in conjunction with Hewlett-Packard's TurboImage[1] database. Through the use of a client/server tool, ALLBASE/TurboConnect[2], and ALLBASE/SQL[3], TurboImage looks like a relational database management system with all the advantages of a relational database.

When a company decides to implement a new system, everyone is affected. The MIS manager will have more responsibilities and will be confronted with new questions each day. The manager will have to ask what changes will need to be made to my current applications? Will all of the users be utilizing the new environment or will my staff need to continue to support the current system as well? Should the new applications be designed to function similarly to the current system? How long will it take to convert the current system and how much will it cost? How will the daily operations of the MIS group change?

The programmer/analyst may be affected even more than the MIS manger. If no one in the MIS group currently possesses the knowledge and skills necessary to develop and support a client/server application, then the appropriate training will be required. Each analyst will also be faced with new challenges and opportunities each day. If the decision has been made to support both the new client/server environment as well as the current system, then what role will I serve in the company? Will both systems need to be changed whenever a user submits a change request? What will I need to learn and how difficult will it be?

---

[1] TurboImage is a registered trademark of Hewlett-Packard.
[2] ALLBASE/TurboConnect is a registered trademark of Hewlett-Packard.
[3] ALLBASE/SQL is a registered trademark of Hewlett-Packard.

Finally, the end user may be affected the most. Whenever a new system is introduced, many end users become cautious and doubt if the new system is really necessary. They may wonder how the new system will affect their jobs. Can all the information which is currently being extracted from the system continue to be retrieved? What new features will be available? And finally, will the new system be difficult to learn?

This paper will review the affects of introducing a client/server environment to each of the previously discussed functional areas (MIS management, programmer/analysts and end users). For the purpose of examples, we have chosen to use GUPTA Technologies' SQLWindows[4] as the client/server software.

MIS Managers

A major change in software must first be analyzed by the MIS Manager to determine if the cost incurred with a new application is offset by the benefits provided. By adding a client/server application that uses the same database as the other applications in the business, the data conversion cost is saved. Once a decision has been made to create a system utilizing client/server technology, a MIS Manager may wonder what changes will be necessary to the current applications. The current applications would not need to be modified. The TurboImage database could be accessed by both the original application and the new client/server application without any modifications, and both systems could be run concurrently if that is desired.

---

[4] SQLWindows is a registered trademark of GUPTA Technologies, Inc.

Another question to be asked is whether the client/server application can be made to look the same as the current system or if this is even desired. In order to answer this, the MIS Manager needs to review the functions being performed by the current application and analyze if the process can be improved by utilizing the technology of Microsoft Windows[5] and the client/server environment. The manager wants to be able to provide more information to the users that can be obtained easier and presented in a more productive manner. Some questions which need to be considered while redesigning a current system are as follows:

Do I want to keep the current system operational or are all of the users going to begin to use the new system?

How can I change the current screens and functionality to

- use multiple windows on a monitor for quicker information access?
- combine several screens that are often used together into a single window?
- utilize scrolling instead of function keys or multiple screens?
- make full use of SQL queries and view information from multiple tables in a single window?

How can I add new functionality by

- utilizing Microsoft Windows Dynamic Data Exchange (DDE) or Object Linking and Embedding (OLE) with other applications?
- creating forms with "What You See Is What You Get" (WYSIWYG) functionality?

As these thoughts take form, the next thing to investigate is the amount of time required for a project of this caliber. Besides installing ALLBASE/TurboConnect on the

---

[5] Windows is a registered trademark of the Microsoft Corporation.

HP3000 and SQLWindows on the PC, the current TurboImage database must be linked to ALLBASE/TurboConnect. After defining what TurboImage database will be linked to ALLBASE/TurboConnect, deciding where the SQL database will reside and determining the sizing parameters, ALLBASE/TurboConnect will create the SQL definition and the data links to the TurboImage database. ALLBASE/SQL can now be used to perform SQL queries against the TurboImage data. ALLBASE/TurboConnect handles the translation of SQL queries to the TurboImage database. The rest of the implementation will follow the typical software development life cycle of design, develop, test and implement. The time period could range from a few days for a simple system to years for something more complex.

Daily operations of the MIS group concerning the HP3000 would not change. Data would continue to be stored in the TurboImage database and backups on the HP3000 would be continued. No additional training would be required in order to maintain the TurboImage database. However, a support role would need to be added for SQLWindows and other PC functions as the user base grows. As a result, an effort would be required to train the MIS staff in ALLBASE/SQL.

Programmer/Analyst

If the decision has been made to maintain both the TurboImage and the SQLWindows systems then the role of a programmer/analyst would be to analyze each change request. This analysis would determine if the modification would affect one or both systems and what changes would be needed. If two separate MIS groups maintain each system, then the modifications would need to be coordinated between the two groups. A change to the database structure would always affect both groups; however, some cosmetic changes may be done on only one system. A decision will need to be made concerning modifications which change the functionality of the system.

When a company decides to convert a system to utilize a client/server technology, the major change which will effect the current analysts will be the need to increase their knowledge base to include a new fourth generation language. The analyst would also need to learn SQL in order to perform queries to the database from the ALLBASE/SQL side.

Even though this is a new environment, SQLWindows is user friendly and makes the transition smooth. The programming environment is point-and-click, with on-line help available. The SQLWindows 4GL is object-oriented, which allows the analyst to define the purpose of a field on a screen without explicitly calling a function to perform the task or using program flow. Screen design is WYSIWYG which makes designing and programming quick and easy.

When designing or developing a new screen, the programmer would choose the item (i.e. an entry field, push button, text, list box, etc.) and place it on the window. That item can then be moved or resized by using the mouse. Double-clicking on any item in the window displays a menu that can be used to define parameters such as item name, field size, format, default text and color.

Using COBOL, paging through a list of records can become difficult because of the need for large arrays, disk files, counters, etc. SQL controls all of the scrolling functions. The records can be displayed page by page using push buttons such as NEXT, PREVIOUS, FIRST, and LAST, or by using a table with scroll bars. SQL programming allows the programmer to use these functions quickly, without writing any interfaces.

End User

Along with the programmer/analyst, the end user will also be affected by the implementation of SQLWindows. Whenever a new system is being introduced, end users tend to become uneasy. Multiple questions flow through their heads:

- Can I access all the information which the current system permits?
- Can I retrieve more information than what the current system allows?
- How difficult will the new system be to learn?

All the information which is available on the current system will still be accessible through the new system. The beauty of converting the current applications to utilize SQLWindows is the availability of the same information in a "friendlier" environment. Figure 1 is an example of an inventory screen created using View Plus and displayed by a COBOL application.

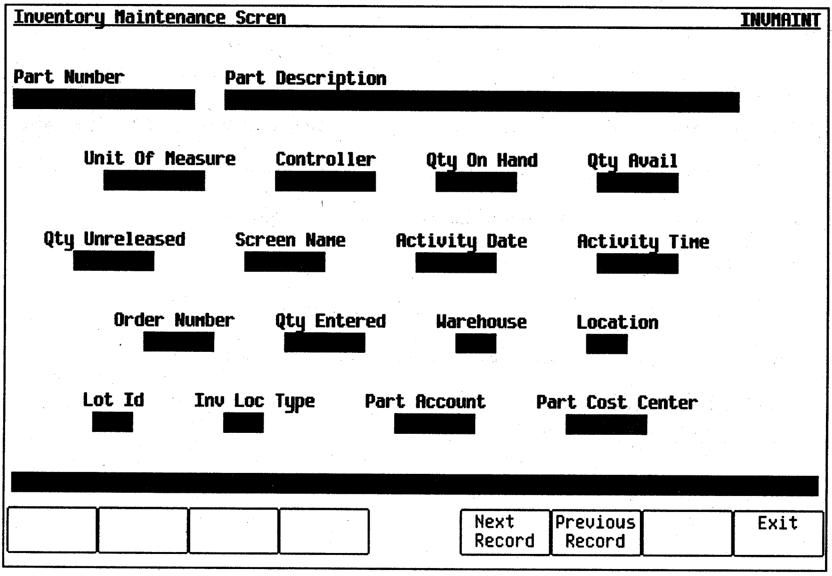

Inventory Maintenance Scren
INVMAINT
Part Number
Part Description
Unit Of Measure
Controller
Qty On Hand
Qty Avail
Qty Unreleased
Screen Name
Activity Date
Activity Time
Order Number
Qty Entered
Warehouse
Location
Lot Id
Inv Loc Type
Part Account
Part Cost Center
Next Record
Previous Record
Exit

Figure 1

COBOL is a good third generation language and View Plus is a good tool for creating screens; however, new technologies make creating screens and accessing data easier and less painful. Imagine the same inventory screen created using SQLWindows. Figure 2 is an example of the same inventory screen created using SQLWindows and utilizing relational database technologies. To many end-users, Figure 2 looks nicer and easier to use than the screen in Figure 1. Also, the screen in Figure 2 gives the user more freedom, which will be discussed later.

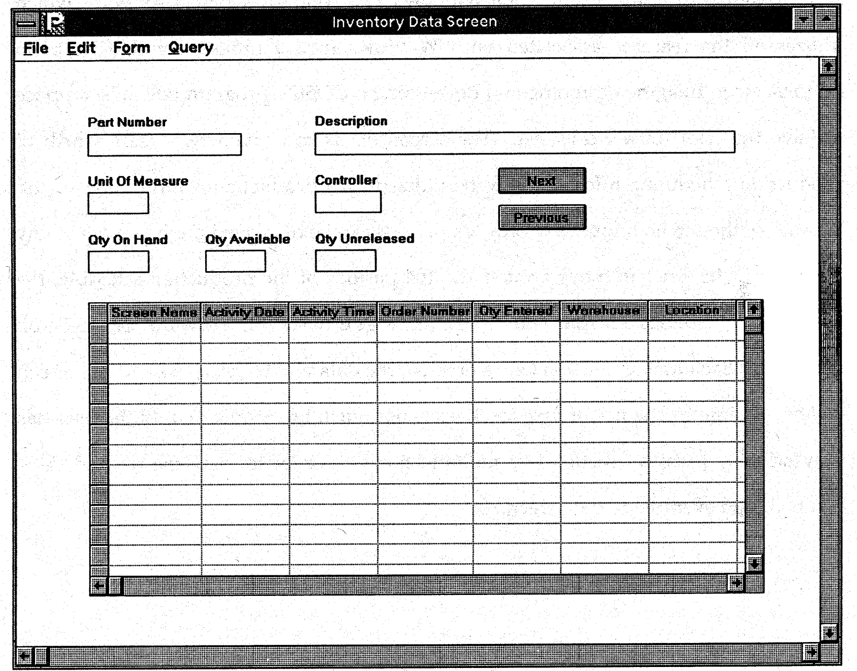
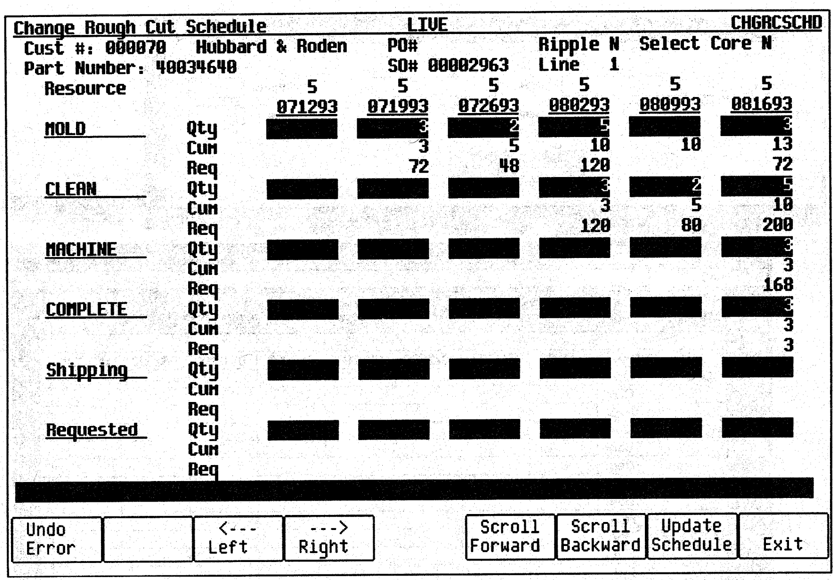

Inventory Data Screen
File
Edit
Form
Query
Part Number
Description
Unit Of Measure
Controller
Next
Previous
Qty On Hand
Qty Available
Qty Unreleased
Screen Name
Activity Date
Activity Time
Order Number
Qty Entered
Warehouse
Location

Figure 2

Many people ask if they will be able to access more information with the new system. Remember, the new system is an enhanced version of the old applications. As previously mentioned, whatever information is available on the current system will be available using the new system; however the decision to allow users to access additional information is up to your company's system administrator.

Another question often asked is how difficult will the new system be to learn. One of the main reasons for converting an existing system is to make the system easier and "friendlier" for the users. SQLWindows is a Windows based software which utilizes all the features associated with Windows (i.e., a mouse, scroll bars, push buttons, etc.). Imagine designing and developing a COBOL program which is suppose to have the "feel" of a worksheet. The screen needs to contain two years worth of production scheduling information to be utilized by a production control department. Because of the screen limitations, only six periods can be displayed on the screen at any given time. In order to move around the 104 periods of the production schedule, the user has to press F3-Scroll Left, F4-Scroll Right, F5-Scroll Forward or F6-Scroll Backward. Each time a function key is pressed, the data will be retrieved and the screen will be repainted. The use of function keys is not much of a problem until the user has to mindlessly press a function key multiple times in order to view the desired data. Figure 3 is an example of this screen.



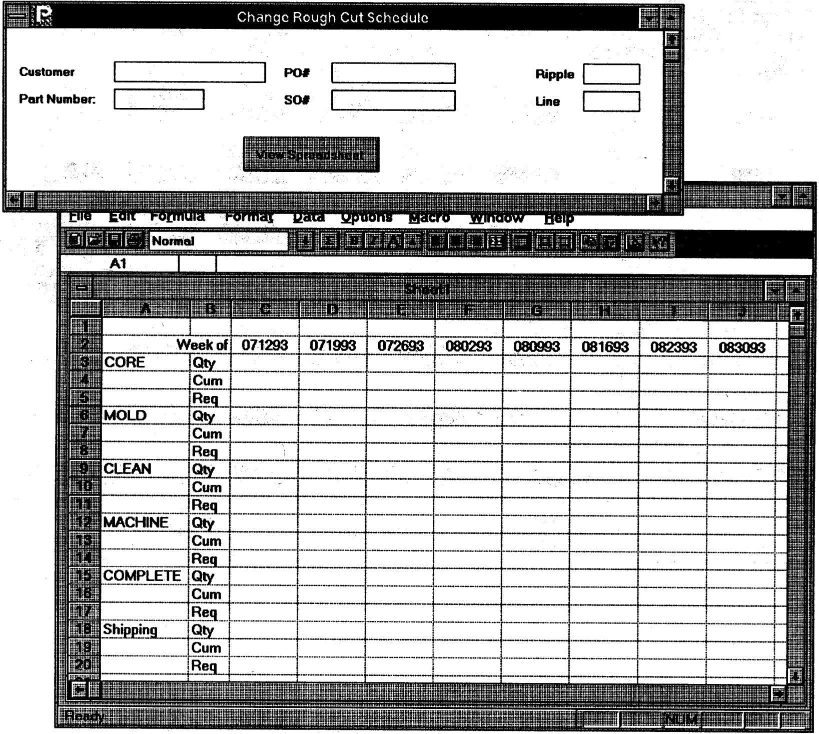
```
Change Rough Cut Schedule              LIVE                              CHGRCSCHD
 Cust #: 000070   Hubbard & Roden      PO#             Ripple N  Select Core N
 Part Number: 40034640                 SO# 00002963    Line   1
   Resource                   5          5          5          5          5          5
                         071293     071993     072693     080293     080993     081693
   MOLD         Qty                      3          2          5                     3
                Cum                      3          5         10         10         13
                Req                     72         48        120                    72
   CLEAN        Qty                                            3          2          5
                Cum                                            3          5         10
                Req                                          120         80        200
   MACHINE      Qty                                                                  3
                Cum                                                                  3
                Req                                                                168
   COMPLETE     Qty                                                                  3
                Cum                                                                  3
                Req                                                                  3
   Shipping     Qty
                Cum
                Req
   Requested    Qty
                Cum
                Req
```

Figure 3

SQLWindows could eliminate the need for the user to have to press one of the four function keys. In this example, the user would enter the key fields and click on the "View Spreadsheet" push button. An actual Excel[6] spreadsheet containing all the production scheduling information will be displayed and the user will have the opportunity to view and/or change the data as desired. Please see Figure 4 for an example of the same production control screen which was created using SQLWindows. To most users, myself included, the screen depicted in Figure 4 would be preferred over the one in Figure 3.

---

[6] Excel is a registered trademark of the Microsoft Corporation.


Change Rough Cut Schedule
Customer
Part Number:
PO#
SO#
Ripple
Line
View Spreadsheet
File
Edit
Formula
Format
Data
Options
Macro
Window
Help
Normal
A1
Sheet1
Ready
NUM

	A	B	C	D	E	F	G	H	I	J
1										
2		Week of	071293	071993	072693	080293	080993	081693	082393	083093
3	CORE	Qty								
4		Cum								
5		Req								
6	MOLD	Qty								
7		Cum								
8		Req								
9	CLEAN	Qty								
10		Cum								
11		Req								
12	MACHINE	Qty								
13		Cum								
14		Req								
15	COMPLETE	Qty								
16		Cum								
17		Req								
18	Shipping	Qty								
19		Cum								
20		Req								

Figure 4

Perhaps the user prefers function keys or push buttons. SQLWindows allows the screen to contain function keys, push buttons, quick keys, scroll bars and pull down menus. Each of which can be included on a screen to suit the user's preference.

Many times an user will need to view all the information pertaining to a specific area, but because of system limitations, multiple screens must be utilized. Now, by using SQLWindows, one large screen can be created containing all the pertinent fields. If the screen is too large and every field cannot be displayed on the monitor simultaneously, the entire screen will actually have scroll bars and all the information can now be displayed on "one screen".

Another feature which is especially useful to end-users is the ability to view multiple screens simultaneously. Because SQLWindows is Windows based, multiple windows can be opened and viewed at one time. For example, an user can enter a receipt of inventory in one screen and instantly view the affect the receipt had on the inventory levels in another screen. This eliminates the need to exit the receipt screen, enter the inventory screen and search for the desired information.

In addition, SQLWindows gives the user the capabilities to copy data directly from an application screen to any Windows based software. Even better, data can be copied from any Windows based software application to a SQLWindows application. Other means exists, DDE and OLE, which will automatically copy data from a SQLWindows application to any Windows based software (and visa versa); however a simple copy feature may be all most users need. This feature will eliminate the need to upload data maintained on the HP3000 to a Lotus[7] or Excel spreadsheet. This in turn, will save the user time, thereby, increasing productivity.

Many businesses today feel entrenched with TurboImage but for a variety of reasons may not want to abandon it. However, they have legitimate business needs which can be solved by using client/server technology while maintaining their TurboImage database. Even though we have addressed a few of the issues which may arise when a company decides to implement a client/server technology, other questions must be addressed and descisions made at each level of an organization.

While we have used SQLWindows as our client package for this paper, it is not the only package available. Any client package that can communicate with ALLBASE/SQL can be used.

---

[7] Lotus is a registered trademark of the Lotus Development Corporation.

5027

PROCESSING AND INTEGRATION OF BUSINESS INFORMATION FROM DECENTRALIZED OPERATIONS

Miguel Cooper
Maria Luisa Bravo
Victor Villalobos

Industrias Resistol, S.A.
Bosque de Ciruelos 99
Bosques de las Lomas
11700 México, D.F.
México
Tel: (525)726-9011

Abstract

A petrochemical business that because of it's geographical distribution in the country, has it's commercial offices in large cities and production facilities far from them and with deficient telecommunications infrastructure. Facilities are built near raw materials and finished products from those plants are sent to other plants where they are used to produce finished goods. The information that flows between plants and commercial offices is needed for production planning and sales. To solve this problem and obtain the desired results in the appropriate time Industrias Resistol implanted a Network to link commercial offices and facilities, and to be able to share, update, and transfer the information, invoices and reports a series of systems have been attached to HP DESK to use it as delivery tool and MAESTRO as control tool. This software have also given us the possibility to define critical processes to be executed automatically assuring the correct execution of them.

INTRODUCTION

In order to keep a business running and in a good position within the market, accurate and fast information is necessary, information about inventories, sales, general ledger, accounts payable and accounts receivable, to know the financial status, and to be competitive giving to the clients the information they request, on the other hand for tax purposes the whole company is seen by the government as one so it is necessary to integrate and consolidate the information for fiscal purposes.

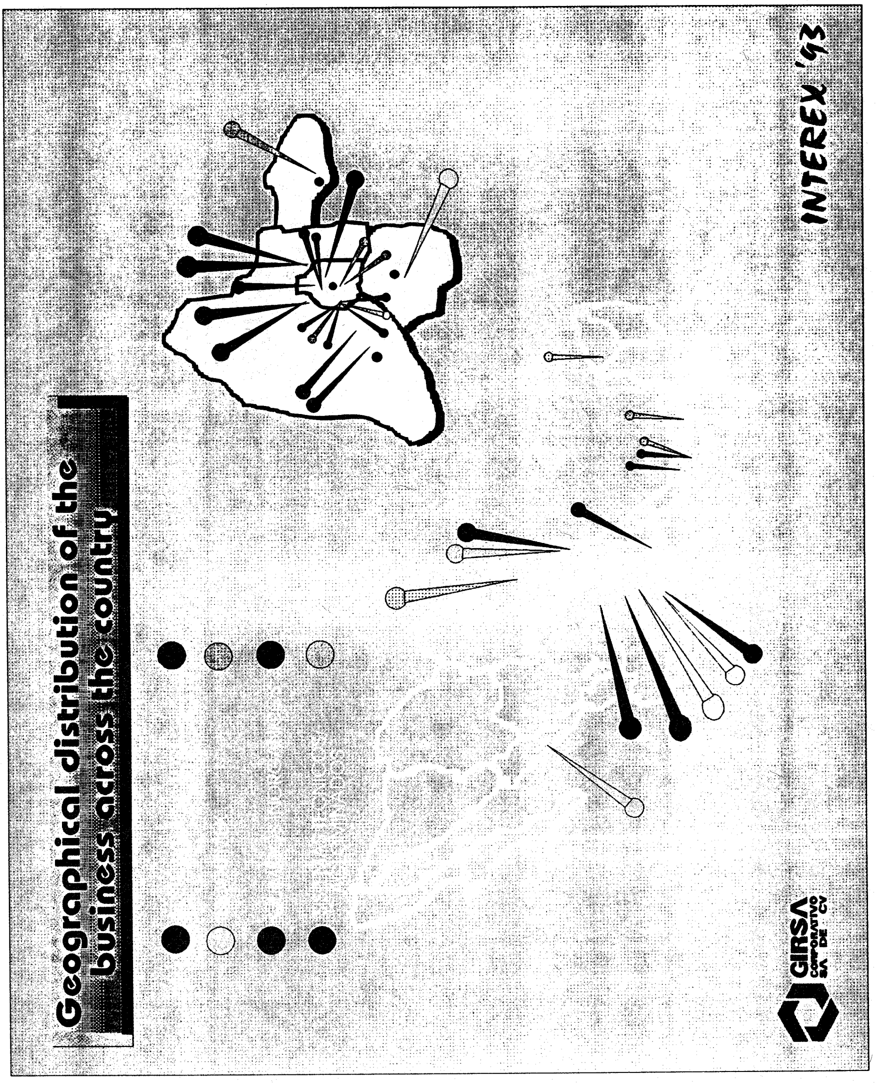
If the business is to be located completely in the same building, the operations would take place in the same area and for hence the systems can be controlled more easily. But today this is not the schema for the big corporations, there are facilities in different locations, there are commercial offices where is considered to be more appropriate, there are sales representatives and external clients that may have the need to access our information. Response time is a very important factor, the hardware, software and communications tools are prepared to handle it, the challenge for the Management Information Systems Area is to introduce all these tools to different areas of the business helping it to be more competitive. (Figure 1)

Geographical Situation and information requirements

Because of the nature of the products manufactured by Industrias Resistol there are facilities that are located where raw materials are produced, in most cases the finished goods produced are used by other facility or a customer to produce other product. On the other hand the commercial offices must be located in large cities where the contact with the customer is needed. Obviously there has to be a good coordination between facilities and commercial offices to be able to produce accordingly to a program and sent to customers the expected quantity.

Industrias Resistol is a large corporation in Mexico its organization is by divisions that handle different markets, for instance the chemical, the commercial, plastics, etc. A factory not necessarily belongs to a division, some give service to two or more divisions. This internal organization added to the geographical distribution of the factories make a very complex data manipulation.

To overcome this situation, first a network should be designed and configured, because of the low reliability in the quality of telephone links, a satellite link was required; more than a centralized operation a concept of decentralization was needed based on the principle that the transaction should be captured where originated and only once, some transactions will affect other locations so a relation was defined and must guarantee that it will arrive to its destination, invoices should be printed where the commercial offices are reducing the time between its generation and payment. All this factors were combined to design a network across the country and contribute to the efficiency of the business as will be explained in the following pages. (figure 2).

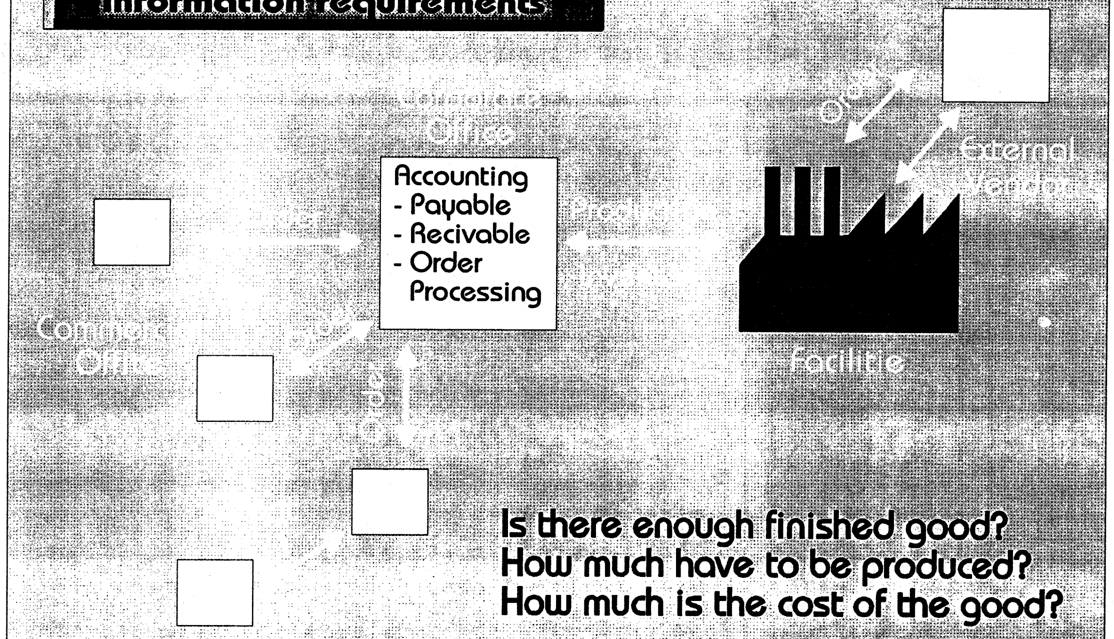


Geographical distribution of the business across the country

INTEREX '93
GIRSA CORPORATIVO SA DE CV

The integration
of business information
Figure 1
5027-3

Geographical situation and information requirements


Corporate
Office
Commercial
Office
Order
External
Vendor
Facilitie

Accounting
- Payable
- Recivable
- Order Processing

Is there enough finished good?
How much have to be produced?
How much is the cost of the good?

GIRSA CORPORATIVO SA DE CV

INTEREX '93

The integration
of business information
figure 2

Once the problem has been identified and it is well understood the relation that must exist between each component of the business, data processing must find a way to provide all the information needed by the different areas of the corporation to keep the business in the market, and these elements are:

- Define the communications requirements (Data and Voice)
- Justify the required equipment: CPU, PAD, Printers, telephonic needs,etc.
- Hardware definition: Capacity, direct and modem ports, communications cards
- Definition of LANs where needed.
- Software evaluation: Operative and applicative
- Training and supervision
- Security
- Performance
- Etc.

These elements must be assembled for each location and for each system on the location, and as the CPU's will be located in different locations a need arose of having a way to verify that all the operations were completed successfully, so a tool to automate and monitor the operation of the whole network must be considered.

THE NETWORK

Because of the existence of a corporate office that among other things consolidate the information to produce financial reports and that the fact that most of the big enterprises have offices in Mexico City a configuration of a STAR Network was chosen, using in most cases satellite links, a central X.25 node and Local networks with Personal computers in some locations. Some remote locations do not have a host computer because they do not process information there, they only access inventories and other systems from a remote site, this locations have PADs to connect asynchronous terminals and in some cases printers defined in some host CPU. This topology is ideal to send information across the network so in some cases the printing is done in a remote location where is needed, the transactions captured in a remote terminal or in a local one, generating in some cases registers that will affect other locations, as it will be seeing.

At the time the network was defined the only link that guarantee near a hundred percent of availability was the use of Satellite links.

The network has grown through the time, new needs have been identified and in locations that initially have only one CPU now have a switch, and a PAD to release the local CPU from processes that only were used as bridges to reach the destination.

THE PROCESSES

This section will describe the processes that run across the network, their evolution and justification, and challenges ahead.

The General Ledger

As established before, Industrias Resistol is a group of facilities and companies that exchange products and therefore information. At the beginning the general ledger system was allocated in a HP 3000/68, the system used near the sixty percent of the total resources of the equipment, the processes took all night to complete in every monthly closing period, the accounting personnel captured, validated and corrected information, people from the distant locations have to travel to Mexico City to capture their transactions, obviously this process was so long that the results of a previous month were ready near the end of the following month. This situation put the business in a vulnerable position because of the lack of real and fast information.

An analysis was made to begin the decentralization of this centralized process, the first step to this approach was to have one micro/3000 for each one of the business, so general ledger of each one will reside on a different CPU but all were physically located in the same site, this solution was not the definitive, many more things should happened first before a complete decentralization could take place. As the systems ran in different CPUs the processes were faster but the process to exchange information was manual, handling many data cartridges. The possibility of errors in selecting the correct catridge was high and the need of traveling from the different locations was still present.

The real decentralization began when the network was created, HP computers were sent to each location, at this time most of them were upgraded to RISC computers, all were configured in the X.25 Network.

The definition of the system was the following: Each location will capture and validate their transactions, a unique general ledger master catalog must exist in each location with the appropriate authorizations to be able to affect accounts of others, the transactions that affect other locations should travel by some way to the destiny in an automatic fashion, in that location this transaction should be applied to the general ledger and validated, the corporate office will receive a confirmation report of all accepted and not accepted transactions. This schema was to reduce the process time, eliminate errors in the application of movements, and reduce the closing periods letting the whole company to have more real and updated information.

As it is seen many aspects must be considered to assure the success of the system:

- The Master catalog must contain unique accounts for each location, and must be updated daily to reflect the last changes.

- A way to ensure the delivery of the transactions across locations.

- A way to coordinate the processes in different locations and to guarantee the correct execution of them.

- A way to ensure that there are the necessary backups to reprocess the information if the need arises.

An important consideration is to keep in mind that the remote locations have different personnel and not everyone have the level of responsability and knowledge required.

Considering these factors it is clear that we needed a tool to coordinate and control the execution of jobs, a tool that let us to transfer reports in automatic to be delivered to the central office, a tool that delivers the information to the appropiate destination in a efficient and automatic way.

After an evaluation of products we selected MAESTRO as the coordination tool, SPOOLMATE to transfer reports and HPDESK as the transport for the transactions.

The solution model works in the following way:

SEND: Transactions are captured where originated, some apply directly to the local general ledger system and some must travel, the control program is the one that select the transactions depending on the account number. Transactions are introduced into traveling files, each one has a letter in its file name that identifies the destination, these files contains the destination name and remittance as HPDESK knows them.

TRANSPORT: A process takes these files to introduce them to HPDESK using its intrinsics, HPDESK activate its TRUCKS to send the messages to the different locations.

RECEPTION: The messages arrive to the Intray area of the destination HPDESK, and for backup purposes a copy of it is automaticaly forwarded to other user. The process of extraction then begins, its first task is to retreive the files from HPDESK and create MPE files, the system then activates the program that takes the transactions into the general ledger of the location, a report is generated showing the status of the operation.

PROCESS SECUENCE AND DEPENDENCIES

The process starts when the CENTRAL MAESTRO launches a job that indicate to the MAESTRO of the locations that the sequence will begin. The first step is to change the Password of the MPE Account to avoid users entering to this system during the execution, the central computer then makes a copy of the changes that apply to the general catalog for each location, when this is done each location takes its copy and update the catalog itself.

After the completion of the catalog a backup of the transactions is done (this phase of the process is in paralell), then the send sequence begins and HPDESK send the messages to the different destinations. To ensure that all the messages arrive to the different locations a dependencie exist, so the extraction of the messages will occur when the send processes in all locations have finished.

Each location then extracts the information and creates MPE files, these files contains all the transactions that will be taken by the process that updates the general ledger Data Base, after each policy is applied a report is generated and send to the corporate office for verification purpouses. The original MPE Account password is returned, and this concludes the daily process.

On fridays a consolidation process is executed before the password is returned, this process consists that on each location a summarization by account number is done, these transactions are sent to the corporate office where are introduced to the consolidated general ledger and as in the previous case a report is generated.

For the closing of the month a process is added before the consolidation, this process handles the exchange rate, this is captured in the corporate office and it is send to the locations, there the revaluation transaction is generated and send to the central office, these are introduced to the consolidate data base and a report is generated.

Previous to the whole sequence some processes are executed to enssure the correct completion of the process, this is a verification that no users are logged in the MPE account, enough disk space is available and that the Data Bases have space to allocate new records.

A diagram of the process is shown in figure 3.

The Factory and the Commercial Office

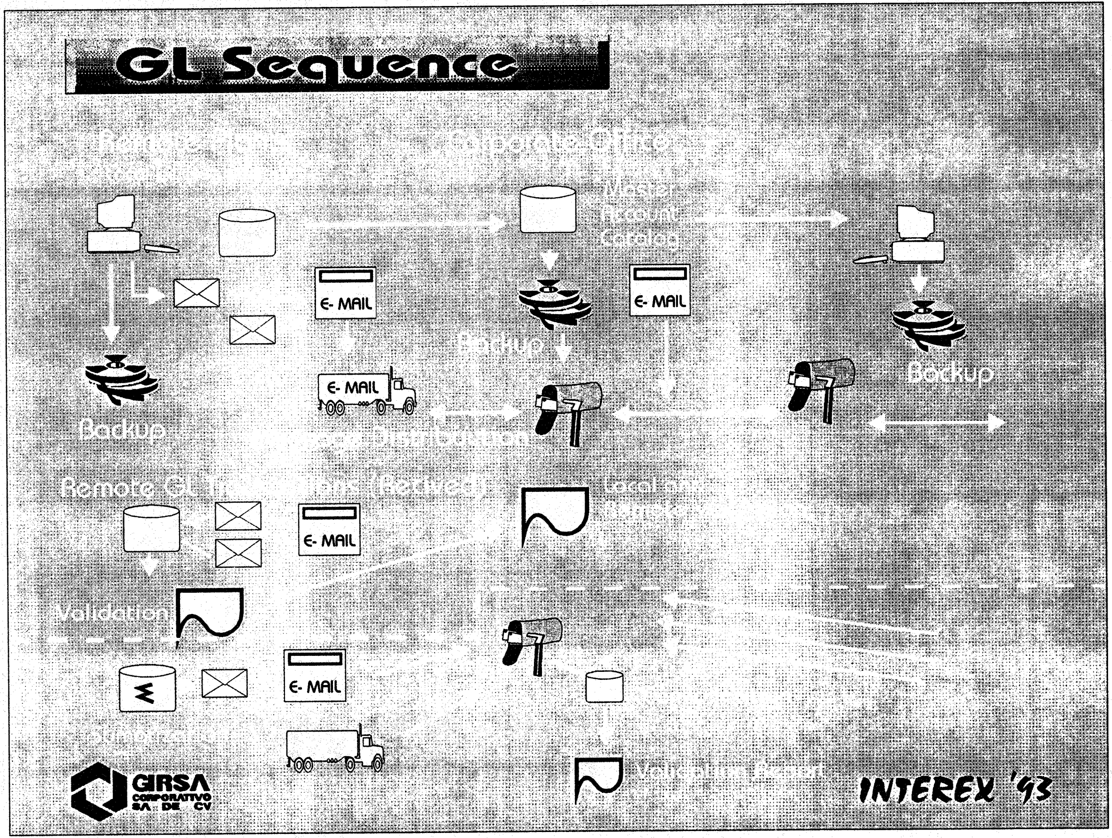
Finished products in facilities most of the times are raw material for a client, as this facilities are located near oil refineries in zones assigned for industrial purpouses as was explained in the beginning of the document, the products are processed and delivered there. The corporate offices of most of the industries are located in big cities, this means that the payment of the invoices has to be done there, the traditional scheme was to generate the document, send it to the main office for registration, so besides the credit time certain number of days have to pass before the customer was able to register it in the account payable system. The cost to financiate this lack of opportunity in the information was very high, so an alternate solution must be found to end with it.

The solution to this situation was to use the Hardware, Software and Telecommunications infrastructure. The product or good is found in the factories where trucks arrive to be filled with it, the factory operator has a terminal and a slave printer to generate the sell note and the invoice is printed in the corporate office where the system reside, this operation let the company deliver the invoices promptly and have the inventories be in their real value (figure 4).

Verification Processes

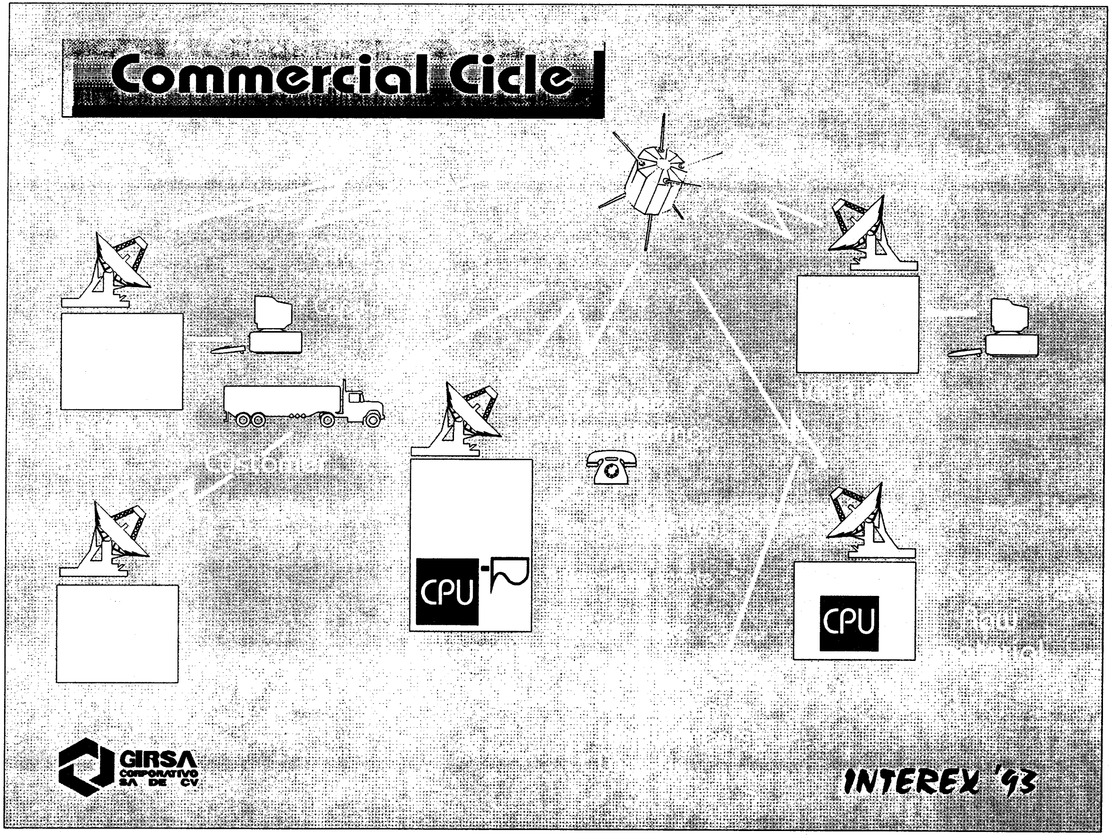
We, as MIS personnel, have to provide to the different business the tools and computer resources to be sure that every process will run to process their information in the appropriate time, for instance we do not want to have full data bases, or lack of disc space messages, for the transfer of information we have to be certain that the HP DESK on each location is up and running, also we have to be sure that the backups have taken place and that the closing processes each day have run. After all the explanation we have done in the previous paragraphs the justification of the automatization tool gave us additional advantages, now we can check in an automatic fashion the availability of disc space and space in data sets, the verification of the HP DESK, the maintenance of it is streamed every week, the partial and full backups are executed by it, some cleaning up processes, production processes are executed and its reports printed in automatic, so the user have early in the morning the information of the daily closing process regardless of his location, this of course let the administrative people take more appropriate decisions in the right time.

GL Sequence


Corporate Office
Master
Account
Catalog
E- MAIL
E- MAIL
E- MAIL
Backup
Backup
Backup
Distribution
Remote GL [illegible] (Received)
Local [illegible]
E- MAIL
Validation
W
E- MAIL
Validation Report
GIRSA
CORPORATIVO
SA DE CV
INTEREX '93

The integration
of business information
Figure 3
5027-9

Commercial Cicle



CPU
CPU
Customer
Raw material
GIRSA CORPORATIVO SA DE CV
INTEREX '93

The integration
of business information
Figure 4
5027-10

THE BENEFITS

The introduction of MAESTRO, SPOOLMATE, HP DESK, DBGeneral as production control tools has given us and the management personnel a total confidence that the information they need will be available in time. The main benefits we have are:

- Automatic Operation (in some processes)
- Availability of resources for critical processes
- Reduction of time in credit to customers.
- Reduction of Time in daily and end of month processes
- Elimination of lost reports.

All these operational benefits plus the economical one as a result of the decrease of personnel displacement, and opportunity of information.

With this schema we have been able to integrate the information of different business that belongs to the same corporation consolidate information from the general ledger and produce Financial reports, knowing the status of each of them without affecting the independence of operation but with the guarantee that all will be executed.

THE CHALLENGES

With the infrastructure mentioned new opportunities are detected each day, now the definition of Local Area Networks in different locations and the possibility to open links between them is a priority, the unattended computer center is an objective to be studied and implemented in the near future, the Client-Server architecture applied to systems that summarize the daily relevant information and present it to the top level executives.

From our private Network we will be changing the IP address to one authorized by INTERNET, this will let us access public networks, the introduction of POSIX in MPE/ix is a first step to migrate to UNIX and think seriously about Open Systems, finally with the electronic mail all across the network the introduction of Multimedia may be a reality soon, the time will come when a customer will fill his requests for materials. All the production cycle will be automatically programmed in accordance of these requests, maybe the charges for the invoices will affect electronically bank accounts. Every business must be prepared to be part of the future, the time when MIS was considered an extension of the GL group has passed, now MIS must be aware of system opportunities to offer a solid base for the decision making process, total quality management and service.

Paper #5030

A Primer on Software Objects

James Wowchuk
Vanguard Computer Services Pty Ltd
72 Moncrieff Drive, East Ryde NSW
PO Box 18, North Ryde, NSW 2113
Australia
Ph +61 (2) 888-9688, Fax +61 (2) 888-3056

Introduction

Who has ever seen a software program without bugs? We're not talking about just a simple routine, but a sufficiently large enough program to be useful on its own. **Does a bug-free program exist?** Probably not. When you buy a program or package do you expect it to work perfectly? Do you take Software Support simply in the hope that they will fix the problems that you'll find in their software? Let's consider the humble personal computer.

Since the introduction of the IBM PC a little more than 10 years ago, they are now found almost everywhere. You could find them controlling business accounts or a virtual reality game. More importantly, one may be controlling a piece of life support equipment in a hospital. Or you might find a PC controlling the security access to a large building complex. You even find PCs on the Space Shuttle. Their role now, can be critical to much life and property.

When you buy a personal computer, you expect at least the following:

- **Quality:** You wouldn't expect the motherboard to start smoking and buzzing. You expect it to all work and continue to work for years..
- **Modularity:** PCs are the sum of their parts...disc drives, monitors, motherboards, adapters. You expect to be able to buy the latest MIDI sound board or video adapter, *not necessarily from the PC manufacturer*, and simply plug it in and have it work
- **Reusability:** You expect to be able to unplug say the video adapter or hard drive in your old machine and put it in your new machine.

If you can't do those all those things, then you might as well have bought an HP150!

So why is it different with software? Surely the design and manufacture of an Intel 80486 chip with a quarter of a million transistor devices must be as complex as a software program. Yet look at the difference in quality! Look at the difference in modularity! How about getting two parts of disparate software programs on the same machine working together? Would you even consider being able to use your WordPerfect *spelling checker* in your Excel spreadsheet? Not likely!

The difference is *objects*. Each PC *hardware object* may be broken down to its components which are in turn other objects. And being objects they are expected to have certain characteristics or behaviour. This behaviour is generally well defined and widely known. Sometimes the object has general characteristics, such as a VGA Adapter card, which may be implemented differently by different manufacturers, but all exhibit the same behaviour. They have dimensions, electrical requirements, temperature operating ranges, and more. People understand these and select those object that are most useful to them.

Now the computer industry is finally coming to terms with ***software objects***. The object oriented principles for systems, languages and architectures are being defined to enable software production to be as dependable, reusable and modular as hardware. More importantly, the *abstractions* of software objects are closer to real life process we are trying to model.

While much has been said of software objects (and we presume that the readers have come across them before), much has been misunderstood.

Common myths about Software Objects

A lot of people already have ideas about object oriented computing and what it may mean. Much of this is may be wrong or mistaken. The old adage "A little knowledge can be dangerous" has never been more true.

Software Objects are for better productivity?

Object oriented computing, in itself, does not specifically change the productivity for development of systems, that is more up to the language and tools used. It is feasible for software objects to be created with Cobol or Basic, but it is not attractive to do so.. Instead the improved productivity associated with software objects is achieved from the *availability of reusable code*, *extensible code*, and *the functional division of code*.

Reusable code is code which can be used by more than one program or system, particularly after the original system is developed. In a lot of ways it is similar to System Libraries (SLs or RLs) containing code which is shared either dynamically or statically. As objects can contain data as well as code, this can be more useful to us.

Extensible code is code (and data) derived from an original class. Where a set of procedures may be needed which is similar to an existing set, rather than re-writing (and possibly duplicating) all the procedures, the derived version only needs to include those ones that are different. The derived class may also include its own data.

Functional division of code results from the grouping of all procedures (methods) and their data into a class definition which is isolated from the principal application. Keeping these procedures together means future enhancements or fixes are easily located and any side-effects minimised.

Software Objects are for systems programming and CASE?

Many software objects are used for systems programming and CASE, but it need not be limited to this. Object oriented computing is a technique which is as applicable to Order Entry as to Network Managers. Some major products that have used software objects are the Next computers operating system, and Novell Netware 4.0, and Windows NT. But to a lesser degree, Speedware uses a form of object techniques with its Designer CASE tool. Vanguard has written a modem terminal access control system for a customer using software objects where Reflection, through its API, was treated as a separate object. Another program we wrote treats all network communication as a software object.

One of the greatest rewards of using software objects is the reduced time for maintenance. As more of a systems/development manager's time is spent in maintaining existing and designing extensions systems, and not on systems programming, software objects can be of greater benefit to them. Individual objects can be easily tested independent of the main program, and validated.

Object oriented computing implies prototyping?

This is true, but not complete. Object oriented computing encourages prototyping but for different reasons than in traditional development. Previously, prototypes were used to "prove" a system design or concept prior to committing the major funds for full development. In a *"mature"* objects environment, there is likely to be a large library of existing object classes to work with. As such, then, these object become building blocks from which it becomes easy to quickly develop new systems. More, though, is the thought that software objects are considered to be dynamic, always changing and evolving as needs require. As such then, it becomes its nature to move directly from prototype to full implementation.

A further consideration is that, using object oriented computing, it is not always necessary to have all the pieces completed to view a prototype of the final result. With software objects we can make simple substitutes for complex objects that have not been implemented yet, much like making a model of a new building using styrofoam and balsa wood. For example, if we needed a complex software object providing a one-way X.25 pipe to an interstate office computer systems, we might substitute with a new related software object that uses a simple file-based structure. The ***interface*** (or visible definitions) remain the same. And the behaviour would mimic (in a limited way) the desired object. Other than this one object class, all other work can continue as if this system was completed. Our simple substitute becomes a *simulation* of the object class we intend to define.

Software objects imply Object Oriented databases?

Typically, we have previously differentiated between data and code. But storing an object means storing data and procedures together. Really this is not that much different than Segmented Libraries (SL) or Relocatable Libraries (RL). What we are saying is that the data and the procedures to modify it are kept together. Remember though an SL can't keep data itself, it can access data through extra data segments. And while the *definition* of an object may be shared, the *instance* or actual object is not.

Users of TurboIMAGE will know that there is a set of routines (which can be known as an Application Program Interface or API) that modify a database. The user data is written to files which while privileged, are still accessible by ordinary file procedures such as FOPEN, FREAD, and FWRITE. Yet we (almost) always use the Image procedures. To do otherwise, we've been taught, is principally wrong, though we have some trust our well-respected third parties, HP and sometimes *our* own internally developed routines. We realise that the strength of our database is not just our data, but the procedures that manipulate them. So we already understand the principal of encapsulation of data. Only those routines may alter our data. But user data is just one of the items of the database. IMAGE has its own internal data structures such as chain pointers, tables, counters, lock descriptors, etc that only the API controls.

But we haven't considered TurboImage to be an Object Oriented database.

In a true object oriented database, not only do we store our data, but we can store our own procedures to manipulate this data. We do this by masking the data definition with a function. Object -Oriented databases are discussed further on.

Object oriented computing demands analysis and design?

There is another saying that Analysis & Design is like foreplay...it can be fun, but it really only gives a yearning for the real thing. All problems should use analysis and design to be solved, but object oriented computing requires a different technique. Object oriented computing tends to works best with continual exchange between the problem domain and the solution domain. As stated before, this is often a closer abstraction, that is a better model, than by other methods.

A good Object Oriented Design methodology will be language independent?

In theory object orientation is an abstraction so there is no specification to the construct of the language. While we initially model on a concept of "objects" during development and even during analysis we generally need to be aware of the constraints or features of the particular language we intend to use. Objects are not a religion. We need to use the parts that are useful to us, and merely be aware of the other constraints.

Principles of Software Objects

Abstraction

So we have some ideas about what object oriented computing is **not**, how about what it is? [1] Firstly, software objects are an ***abstraction***. They attempt to take the real world and model it with a *simple* system which mimics it. Abstraction is a natural part of the mental behaviour of people. It allows you to remove yourself from much of the detail in order to focus on areas that are of interest.

☞ Moving from machine language to Assembler was an abstraction that changed the necessity of programmers to know bit patterns, to knowing the *purpose* of the patterns. The compiler held the *knowledge* needed to create the bit patterns from the symbols.

☞ Moving to higher level programming languages like Cobol or Fortran was an abstraction which removed much of the machine specific or architectural details from programs. Again the local compilers contained the knowledge needed for that particular machine or architecture.

☞ Structured programming was an abstraction that encouraged standard handling of *looping* and *if-then-else* conditions found in the higher level languages. This removed much of the details related to changing execution sequences. This knowledge remained with the entry-level programmers who wrote the actual code, or in CASE tools.

☞ Software objects are an abstraction that removes the detail of specific problem-solving code and encapsulates it *within* the object. Now each object contains the *knowledge* needed to perform their own problem-solving role.

Changes in Thinking

For analysts working with software objects, one requires a shift in the thinking and attitudes. The change is from the traditional *procedural* view to one of *object-oriented*. The suggestion that one can take an overall view of what a system does is flawed in many practical cases. For example, how do you provide a meaningful overall view of a multi-processing operating system? Complex tasks can not be broken down simply by their functions. The purpose of a system is not so clear. As a result the development of functions may pass through many layers of the functional decomposition. The analyst is typically encouraged to focus on an algorithm early on, then work on a analysis to support the algorithm. If this algorithm should prove flawed, then the whole of the analysis will need to be reconsidered.

For the *object-oriented* thinking, the first step is identifying object which when assembled can form a system. This is the first step in encapsulating into objects, then studying the relationships between these objects. Once a model is reached, then the behaviour of each object can more closely studied and defined. The actual algorithms used to develop this behaviour are not done very late in the development process. This means that if the algorithm should prove flawed, that alternatives can be considered that which won't drastically affect other objects in the system.

---

[1] While there is much in common amongst the various proponents of software objects, the terms tend to vary greatly. The term *Class*, and *Objects* are particularly open to interpretation.

Procedural Thinking	Object-Oriented Thinking
What does the system do?	What objects make up the system?
What is its purpose?	How do the object relate to each other?
How do I design the code to achieve the functional behaviour?	What is the behaviour of each object?
Focus on algorithms at start.	Algorithm functions deferred to object coding.

Changes in Culture

The acceptance of object oriented techniques requires a change in the management culture to accept it. Most companies have grown up with a *project* culture. This is typified by looking for specific results to achieve immediate profits. A project is typically undertaken within a single company department, and has short term perspective, looking to the immediate implementation of the project. It may be composed from various programs and the design is done from top down. The languages of Cobol, Fortran, Pascal and C are the most popular for implementing a project. It typically pays little heed to the longer term maintenance of the project, or future enhancements. When the project is implemented, the project is complete.

The *product* culture does not generally look for a payback in their first implementation of software objects. Instead its benefits come from the longer term, such as further enhancements, and reduced maintenance time. Libraries are built-up of objects, or components defined and tools created that are applied for the entire company, or even industry wide. The development of systems is then possible by the now easier assembling of components. It is a longer term viewpoint, which recognises that while the product exists, the process is not closed. Naturally this requires more bottom-up development strategies. While many of the techniques of software objects can be done with traditional languages, the benefits of abstraction are more easily achieved using object-oriented languages such as C++ and SmallTalk.

Characteristics	Project Culture	Product Culture
Method	Functional	Object-oriented
Outcome	Results	Components, Libraries, tools
Economics	Profits	Investment
Unit	Department	Company, industry
Time frame	Short-term	Long-term
Goal	Programs	Systems
Bricks	Program elements	Software objects
Strategy	Top-down	more Bottom-up
Language	Cobol, Pascal, C, etc	C++, SmallTalk, Eiffel, etc

Definitions used with Software Objects

State - data with particular values

Behaviour - methods

Class - A definition of an object. Objects which have the same behaviour are said to belong to the same class. It is the template used to create the object. Its like the Cobol source code to a program, it defines the behaviour of the program but in itself can do nothing.

Instance - When an object is created , it then has its own state and behaviour. This will be different than any other object, even of the same class. For example when each person runs a program they each have a different instance. While they may share the same program code, they have their own stack markers (program state), and their own data segments (data state).

Specification (Abstraction)

Message

The *message* is what the **calling** program (or object) issues. It consists of a name and parameters. It can be considered similar to a procedure or function call.

Signature

The *signature* is the public definition or specification of what an object offers. A compiler of a calling program will need to check the signatures to determine if the message being issued is valid. The signature doesn't contain code. On the HP3000 this is similar to the SPLINTR files which validates intrinsic calls to procedures.

Method

The *method* is the actual code invoked when an object receives a message.

Encapsulation

Encapsulation may be defined as the taking a record of data and combining with it the functions and procedures to manipulate it. This conceptually draws a capsule or barrier around these bundle. It aids in our semantic ability to deal with the enclosed complexity. While there may be many procedures or methods used to manipulate the data, not all of them need be visible. Within the encapsulation we may hide some of the data and procedures, that is removing them from the public interface. *Information hiding* distinguishes the ability to perform some act from the specific steps and information needed to do so.[2]

Inheritance

Inheritance is another abstraction mechanism where the definition of an object class is descended from a one or more other object classes. All common data and procedures are inherited, with new procedures available to overwrite or replace inherited ones. This allows us to conceive of the new object as a refinement of another. We abstract out the common features and focus on the differences. It allows us to invoke the same behaviour without re-inventing it each time. We gain code re-use through inheritance.

For example, I could define an object class called **BANK ACCOUNT.** It would contain signatures to OPEN, CLOSE, CALC_INTEREST, DEBIT, CREDIT . It may or may not define methods for all these signatures. But it could handle a lot of the common work needed to operate an account.

Now we would need to define subclasses of BANK ACCOUNT such as **PASSBOOK**, and **CHEQUE**. As subclasses, these new objects inherit all the methods and data structures defined for BANK ACCOUNT. For the PASSBOOK class, we would need to redefine the CALC_INTEREST method to meet with the rules that it has. For the CHEQUE class, though, the CALC_INTEREST method would do nothing...we aren't paying interest on cheque accounts. But we may wish to change the DEBIT method for this class to add a charge for each cheque debited.

The net effect is that these subclasses only contain the code which is different from their class which they inherit from.

Polymorphism

The ability to invoke the same method name in two different objects is known as polymorphism. The use of a common predecessor class and function which when implemented uses the class and function of the descendant class created. This permits extendable code. For example, in the BANK ACCOUNT example, we may write a routine that receives a BANK ACCOUNT object to which it will apply the method CALC_INTEREST. If the routine is passed a CHEQUE object, then it will apply the CALC_INTEREST method that belongs to the class CHEQUE. If passed a

---

[2] Rebecca Wirfs-Brok, Brian Wilkerson and Lauren Wiener: ***Designing Object-Oriented Software***, PTR Prentice-Hall, Englewood Cliffs, New Jersey, 1990, pg 19

PASSBOOK object, then it will apply the CALC_INTEREST method for PASSBOOK. Yet, when this routine is written, it does not know nor need to know that there are subclasses with their own definitions. The object environment will automatically select the correct method for the object class.

Benefits of Software Objects

- √ **CORRECTNESS** - software objects do what they are designed to do.
- √ **ROBUSTNESS** - the ability of software objects to handle what they weren't designed to do. Reliability.
- √ **EXTENSIBILITY** - the ease in which software objects may be adapted to changed specifications.
- √ **REUSABILITY** - the ability to use software objects in new or different applications. Modularity.
- √ **COMPATIBILITY** - the ease in which software objects can be assembled.
- √ **EFFICIENCY** - the ability to design software objects to make the most effective use of the system resources available to it.
- √ **PORTABILITY** - changes that are platform dependant are more easily identified and isolated within a system.
- √ **VERIFIABILITY** - individual objects are more easily proved during testing, or if needed during debugging.
- √ **INTEGRITY** - the breakdown of a system into component objects means that security to specific sensitive objects can be made more restrictive, without limiting programmer access.

Development strategies

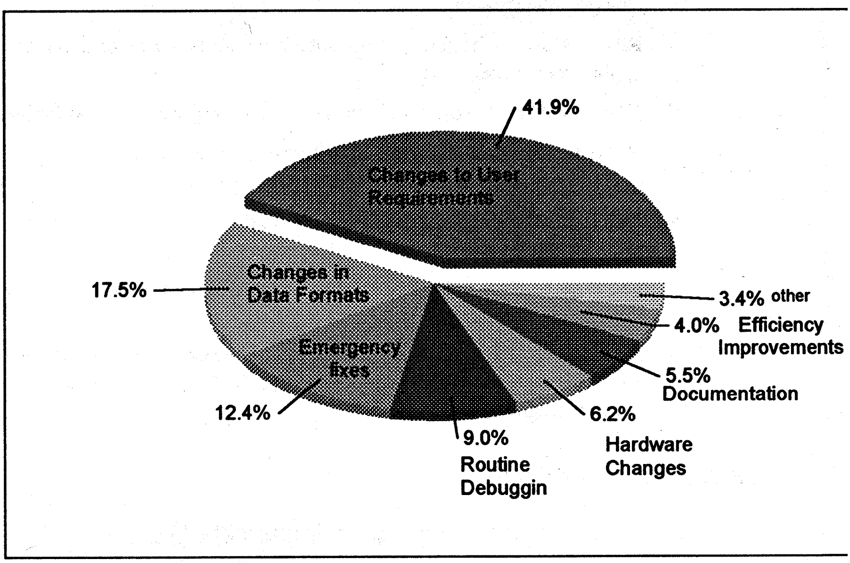
Traditional Analysis -> Design -> Implement (ADI)

Most major systems development is accomplished using the methods that have been described as the waterfall method or the spiral method. This typically involves reaching specific stages of Analysis, Design and Implementation before the "next" stage can begin. There can be little overlap as all the bottom pieces must be complete to form the next higher piece. Only when a the top is reached can the next stage proceed. Most analysis on systems done today uses some type of this top-down design. The most popular are Yourdon's "bubble diagrams" and DeMarco's Functional Decomposition.

Bertrand Meyer[3], developer of the Eiffel object language, identifies four flaws in this top-down functional design:

1) It fails to account for the evolutionary nature of software systems.
2) Questionable notion of a system characterised by a single function.
3) It is a functional mindset; the data structure aspect is often neglected.
4) Does not promote reusability

Traditional ADI is based on new system development. It caters little for maintenance. Various studies have estimated that 70% or more of the total cost of software is devoted to its maintenance, yet this method ignores this particular focus. A study by Lientz[4] revealed the breakdown of these costs in Maintenance.



OOPSADI

OOPS limits ADI to smaller parts (clusters[5]) giving smaller teams and tighter controls. A small team does all three A.D.I. Project Manager splits into clusters. Each cluster can be follow the ADI on its own, generally independent of the others. This means, more often than not, that the same people can be doing this task. This continuity provides even better stabilisation of the end product.

---

[3] Betrand Meyer, *Object-Oriented Software Construction*, Prentice-Hall, Hemel Hempstead, 1988, pg 44

[4] BP Lientz, Software Maintenance: A User/Management Tug of Ware, *Data Management*, Apr 1979, pp 26-30

[5] Betrand Meyer, Writing correct software, *Dr Dobb's Journal*, Dec. 1989, pp 48-63

Object Oriented Programming Languages

C++

In 1983/84, Bjarne Stroustrup of AT&T, created "a better C" language. These extensions to Brian Koenig and Dennis Ritchie's C language (also of AT&T) implemented various object oriented facilities among others. The X3J16 committee of ANSI (convened by HP in 1989), is expected to have a standard out by next year . Available on more platforms than any other object language, including HP9000s, it is expected to soon be available on the HP3000 systems. PC versions are available now that support most of the current Version 3 (as defined by BS) features though perhaps not all, especially exception handling.

As with C, there are those that love it, and those that hate it. Some of the features, ostensibly to provide either performance or just to get things working, run foul of software object purists. In some ways it violates many of the characteristics. But the variety of implementations, especially on the PC platform, mean a continual changing environment. This is resulting in a more and more "pure" object language which is still capable of the performance and intricacy found in C.

Objective C

A different approach than Stroustrup, Brad Cox of Productivity Products International grafted some SmallTalk concepts onto the C base. It focuses strongly on the dynamic binding (and resulting polymorphism), in a static environment. This means it can be compiled to provide a small easily implementable structure, such as a program.

SmallTalk

Developed by the Xerox Parc group in 1976, by Adele Goldberg and Daniel Ingalls, much of the original work was done by Alan Kay about 1970 while a graduate student at the University of Utah. Kay later worked for Xerox, under Adele, where he used these principles as the basis of an advanced micro-computer system. From these systems, windows, graphical user interfaces and even some concepts of Artificial Intelligence emerged. SmallTalk holds the largest share of the object-oriented market and is available from a number of suppliers and on a number of platforms.

SmallTalk is pure object-oriented. Everything in the environment of SmallTalk is an object. This includes numbers, strings...everything. Everything descends from the base class known as OBJECT. The definitions and code for objects are themselves instances of objects under OBJECT. This means the environment for developing is an object, libraries are objects and even debugging tools and classes are objects.

SmallTalk depends on dynamic binding (rather than compiled *static* binding). Only when the message is sent to the subject object is validity checked. This means that changes to object classes can be done even on a running system. This is similar to an interpretive environment. As well, since classes are objects, *class routines* can be defined which are applicable to all objects rather than a single instance. For example, a counter that gets incremented each time a certain method is invoked could be done by class routine. The resulting count is not just for that instance of the object but for all objects of that (or descendant) class.

There is little capability for persistence (saving objects) within SmallTalk. Instead, you need to rely on saving the entire environment to preserve the system state. Specific methods may be written for saving and restoring specific instances.

SmallTalk does not support a single class having two parents (superclasses). A class can have one and only one class it is descended from.. Garbage handling (instanciation and destruction) is automatic.

SmallTalk is considered useful for teaching concepts but difficult for large systems.

Eiffel

Another *pure* object-oriented language, the languages is derived from Simula and Ada. Developed by Bernard Meyer, it can generate either its own object modules, or 'C' code for portability. It has limited ability to provide persistence on its own. It provides its own environment for development, and provides large libraries of standard object classes for handling windows, database and other bases operations. As an academic, Meyer has chosen to provide a product that meets the latest definitions of object oriented computing. This includes support for the latest concepts such as "contracts" .

Simula

Simula is one of the earliest published examples of an object-oriented languages. It was developed in 1967 by Ole-Johan Dahl and Krysten Nygarrd from the University of Oslo and the Norwegian Computing Centre. An object-oriented extension of ALGOL 60, it still is based on classical procedural coding, with only limited modularity or separate class compilations. Most implementations of the language are in Norwegian or Swedish.

Ada

Ada may not be considered a full object-oriented language, as it has no support for inheritance. But its use for encapsulation and operator overloading brings it close. A version known as "Classic Ada", has implemented many OO features. Proposals to introduce object-orienting into Ada has been under discussion with the US Dept of Defence for the last few years.

Object Cobol[6]

An interesting development is the proposal (again by HP) in 1989 of an Object Oriented Cobol. Currently it is being considered for committee of ANSI and ISO (Codasyl). The attractiveness of the proposal is its ability to take advantage of the wealth of Cobol code (over 70 billion lines!) already in existence. The features of software objects would then be readily applied to this code resulting in more modular, and reusable with higher quality.

The extensions proposed include capabilities for Class Definition, Inheritance, Signatures (interface definition), static and dynamic binding (Polymorphism). The following code is an example of Cobol class definition being proposed.

```
IDENTIFICATION DIVISION
OBJECT-CLASS.  BANK-ACCOUNT
DATA DIVISION
WORKING-STORAGE SECTION.
01 BALANCE                       PICTURE $$$,$$$.99 VALUE ZERO
01 ACC-NUMBER                    PICTURE 9(14).
01 CUST-INFO.
   05 CUST-NAME                  PICTURE X(20).
   05 CUST-SS-NUM PICTURE X(9).
PROCEDURE DIVISION.              /*** object methods declared here.

IDENTIFICATION DIVISION.         /*** 1st method declaration.
METHOD-ID. RETURN-BALANCE IS PUBLIC.
DATA DIVISION.                   /*** method local data declared here.
LINKAGE SECTION.                 /*** formal parms
01ACCT-NUM                       PICTURE S9(14).   /*** parm
01ACCT-BAL                       PICTURE $$$,$$$.99         /*** return val
PROCEDURE DIVISION
     USING ACCT-NUM RETURNING ACCT-BAL./*** method prototype
     MOVE ZEROS TO ACCT-BAL.
     IF ACC-NUMBER = ACCT-NUM
        MOVE BALANCE TO ACCT-BAL.
     EXIT METHOD RETURNING ACCT-BAL.
END METHOD RETURN-BALANCE.                /*** Method code ends here

IDENTIFICATION DIVISION.         /*** 2nd method declaration
METHOD-ID. WITHDRAW IS PUBLIC.
/* withdraw from account; code not shown
END METHOD WITHDRAW.

IDENTIFICATION DIVISION.         /*** 2nd method declaration
METHOD-ID. DEPOSIT IS PUBLIC.
/* deposit in account; code not shown
END METHOD DEPOSIT.

IDENTIFICATION DIVISION.         /*** 2nd method declaration
METHOD-ID. PUBLIC IS PUBLIC.
/* print account info; code not shown
END METHOD PUBLIC.
```

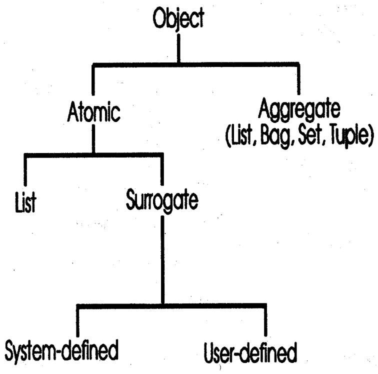
---

[6] Megan Adams & Dmitry Lenkov, Object-Oriented COBOL: Then Next Generation, *Proceedings of the Interex HP Users Conference*, New Orleans 1992, Volume 2, paper 3864.

Object Oriented Databases

A different approach to the paradigm is the use of object oriented databases. These databases store, find, share, manipulate and protect objects, that is data plus methods, instead of just data alone. This abstraction removes the need for individual programs to know how to manipulate objects, and instead permits programs (both procedural and object-oriented) access to the knowledge kept by the database. It is in effect a object oriented shell around a standard RDBMS (such as Allbase) bringing much of the features of object orientation to conventional programs. A chief benefit, is that it brings a diverse range of products to working together through the centra database.

OpenODB from Hewlett-Packard is offered on HP3000 and HP9000 systems.



Object SQL combines the rules of object-oriented programming (abstraction, encapsulation, extensibility and reuse) with the features of RDBMSs (declarative queries, authorisation and views).

Distributed Object Management Facilities (DOMF)

One of the real futures for software objects lies in the area of Distributed Computing. Objects offer a much closer abstraction for handling the complexities involved.

Object Management Group

Distributed Object Management

Object signatures defined in IDL

IDL compiler generates stubs and type repository

Object creation and deletion

Object class installation

Distributed Object Services

Object location services

Automatic object activation and deactivation

Dynamic and static method invocation

Common User Services

Security services

Management of persistent state

Management of object information

Persistent references between object

Event notification service

Interface services to facilitate task automation

Application Objects

DAA objects combined to form a solution

Applications may be separately developed

DOMF provides integration between applications

Services provided to "embed" existing applications

Conclusion

Some have argued that object oriented systems are merely redefined Structured programming examples. I believe this overlooks many of the chief strengths software objects promise. Firstly, with the ability to independently test objects out of their context, it offers the promise of reduced errors, and easier maintenance. Secondly, the ability to redefine and change existing objects, by merely writing the changes you want promises better interfacing with the end user. The concepts of object oriented software are being felt in everything from electronic mail, through user configurable "Speedbars" in word processors and spreadsheets.

There is still a lot of work to accomplish before software objects become universal. There will be much activity and changes coming. But the winners will be those who adopt the technology early and are better able to adapt to the changes later.

PAPER NUMBER: 5031

TITLE: English 1A -- A DP Professional's Guide to Writing

PRESENTER:

Pamela Dickerson
ECI Computer, Inc.
P.O. Box 16834
Irvine, CA 92713
714-434-8841

HANDOUTS WILL BE AVAILABLE AT TIME OF PRESENTATION.

Interex

P.O. Box 3439
Sunnyvale, Ca 94088-3439
(408) 747-0227
Fax (408) 747-0947